

(12)特許協力条約に基づいて公開された国際出願

(19)世界知的所有権機関
国際事務局(43)国際公開日
2004年4月22日 (22.04.2004)

PCT

(10)国際公開番号
WO 2004/034698 A1

(51)国際特許分類7: H04N 5/44, 5/93

(21)国際出願番号: PCT/JP2003/012932

(22)国際出願日: 2003年10月9日 (09.10.2003)

(25)国際出願の言語: 日本語

(26)国際公開の言語: 日本語

(30)優先権データ:
特願2002-296234 2002年10月9日 (09.10.2002) JP

(71)出願人(米国を除く全ての指定国について): 松下電器産業株式会社 (MATSUSHITA ELECTRIC INDUSTRIAL CO., LTD.) [JP/JP]; 〒571-8501 大阪府門真市大字門真1006 Osaka (JP).

(72)発明者: および

(75)発明者/出願人(米国についてのみ): 堀井 幸

(HORII,Yuki) [JP/JP]; 〒567-0803 大阪府茨木市中経寺町5-4-305 Osaka (JP). 川上 義雄 (KAWAKAMI,Yoshio) [JP/JP]; 〒571-0079 大阪府門真市野里町6-16-201 Osaka (JP). 脇 康 (WAKI,Yasushi) [JP/JP]; 〒619-0232 京都府相楽郡精華町桜ヶ丘1-30-16 Kyoto (JP).

(74)代理人: 小笠原 史朗 (OGASAWARA,Shiro); 〒564-0053 大阪府吹田市江の木町3番11号 第3ロンヂエビル Osaka (JP).

(81)指定国(国内): CN, JP, US.

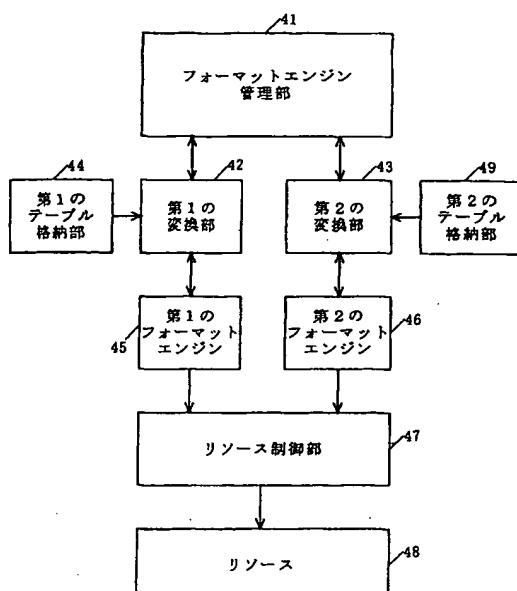
(84)指定国(広域): ヨーロッパ特許 (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HU, IE, IT, LU, MC, NL, PT, RO, SE, SI, SK, TR).

添付公開書類:
— 国際調査報告書

/統葉有/

(54) Title: INFORMATION PROCESSOR

(54)発明の名称: 情報処理装置



41..FORMAT ENGINE MANAGING SECTION
42..FIRST CONVERTING SECTION
43..SECOND CONVERTING SECTION
44..FIRST TABLE STORING SECTION
45..FIRST FORMAT ENGINE
46..SECOND FORMAT ENGINE
47..RESOURCE CONTROL SECTION
48..RESOURCE
49..SECOND TABLE STORING SECTION

(57) Abstract: An information processor in which addition or alteration of a format engine can be dealt with readily. The information processor comprises a format engine managing means, and an operation control means. The format engine managing means previously defines a common state where the operating state of the format engine is defined by a representation common to all format engines, and manages the operation of each format engine. The operation control means previously defines the correspondence between the common state and the individual state where the operating state of the format engine is defined by a representation different for each format engine and controls the operation of the format engine such that an arbitrary individual state is brought about.

(57) 要約: フォーマットエンジンの追加や変更に容易に対応することができる情報処理装置を提供する。情報処理装置は、フォーマットエンジン管理手段と、動作制御手段とを備えている。フォーマットエンジン管理手段は、フォーマットエンジンの動作状態を全てのフォーマットエンジンに共通の表現によって規定した共通状態を予め定義しておき、各フォーマットエンジンの動作を管理する。動作制御手段は、各フォーマットエンジンに対応して設けられ、共通状態と、フォーマットエンジンの動作状態を各フォーマットエンジン毎に異なる表現によって規定した動作状態である個別状態との対応を予め定義しておき、任意の個別状態となるようにフォーマットエンジンの動作を制御する。

WO 2004/034698 A1



2文字コード及び他の略語については、定期発行される各PCT gazetteの巻頭に掲載されている「コードと略語のガイダンスノート」を参照。

明細書

情報処理装置

技術分野

本発明は、情報処理装置に関し、より特定的には、JavaやHTML等の特定のフォーマットで記述されたアプリケーションを実行するフォーマットエンジンの動作を管理する情報処理装置に関する。

背景技術

近年、デジタルテレビ、携帯情報端末、携帯電話といった情報処理装置において、JavaプログラムやHTMLコンテンツ等の特定のフォーマットで記述されたアプリケーションを実行することが盛んに行われている。ここで、このようなアプリケーションの実行やデータの再生を行うためのソフトウェアをフォーマットエンジンと呼ぶ。フォーマットエンジンとは、例えば、上記Javaプログラムを実行するためのJavaミドルウェアやHTMLコンテンツを表示するためのブラウザである。これらのフォーマットエンジンの他にも、現在パーソナルコンピュータ上で実行されているPDFデータや音楽データ等を再生するためのフォーマットエンジンも、上記デジタルテレビ等において実行されるようになるであろう。

一般的に、上記デジタルテレビ等の情報処理装置に格納されているフォーマットエンジンは1つである。図94は

、従来のデジタル放送受信機の構成図の一例である。このデジタル放送受信機は、放送データに含まれるアプリケーション（Java プログラム）を実行し、管理するものである。図 94において、デジタル放送受信機 9402 は、テレビモニタ 9401 に接続されている。また、デジタル放送受信機 9402 は、ディスプレイマネージャ 9411 と、アプリケーションマネージャ 9412 と、ルール 9414 と、信号モニタ 9413 とを備えている。

図 94において、デジタル放送受信機 9402 の信号モニタ 9413 は、放送データストリームを受信する。信号モニタ 9413 は、放送データストリームに含まれるアプリケーションが含まれるか否かを判断し、アプリケーションの有無を示す信号データをアプリケーションマネージャ 9412 へ出力する。アプリケーションマネージャ 9412 は、入力される信号データに基づいて、放送データストリームにアプリケーションが含まれるか否かを判断する。そして、放送データストリームにアプリケーションが含まれる場合、アプリケーションマネージャ 9412 は、当該アプリケーションをアプリケーションを読み込み実行する。アプリケーションが実行されると、テレビモニタに表示されるべきディスプレイ情報がディスプレイコンテキストとしてディスプレイマネージャ 9411 へ提供される。ディスプレイマネージャ 9411 は、適切なディスプレイ情報をテレビモニタ 9401 に提供する。ここで、アプリケーションマネージャ 9412 は、ルール 9414 に従つて動作し、各アプリケーションの状態をロード状態、一時

停止状態、アクティブ状態、および抹消状態のいずれかに遷移させる。ルールは、例えば、「1度に1つのアプリケーションのみをアクティブ状態にする」や、「1度に1つのアプリケーションのみを表示できる」等である。デジタル放送受信機9402は、このようなルールを用いることによって複数のアプリケーションを管理することができる。

一方、デジタルテレビ、携帯情報端末、携帯電話といった情報処理装置が複数のフォーマットエンジンを備える場合、それら複数のフォーマットエンジンを管理する技術も考えられている。このようなデジタルテレビ、携帯情報端末、携帯電話といった情報処理装置では、CPUに比べるとフォーマットエンジンを実行する際に用いられるリソースが限られているという側面がある。リソースとは、例えばメモリのように、フォーマットエンジンによって利用されるハードウェアをいう。例えば、デジタルテレビが有するリソースには、メモリの他、放送波を受信しデータを復号するためのチューナ、放送波に含まれる映像や音声をデコードするためのデコーダ、インターネットに接続するためのネットワークインターフェース等がある。フォーマットエンジンを実行する際には、実行されるフォーマットエンジンが利用するリソースを確保する必要があるので、上記のようなリソースが限られた情報処理装置においては、各フォーマットエンジンの実行を管理することが行われている。例えば、デジタルテレビでは、各フォーマットエンジンを切り替えて実行する処理を行うものがある。以下、

従来のデジタルテレビにおいて複数のフォーマットエンジンを管理する技術を説明する。

図95は、従来のデジタルテレビにおいて用いられるミドルウェア切替装置の構成図の一例である。このミドルウェア切替装置は、複数のミドルウェア（フォーマットエンジン）を切り替えて実行する。図95において、ミドルウェア切替装置は、ミドルウェア決定部9501と、ミドルウェア読込部9502と、ミドルウェア実行部9503と、ミドルウェア格納部9504とを備えている。

図95において、ミドルウェア格納部9504は、第1～第nのミドルウェアを格納している。ミドルウェア決定部9501は、第1～第nのミドルウェアの内、実行すべきミドルウェアを決定する。ミドルウェア読込部9502は、ミドルウェア決定部9501によって決定されたミドルウェアをミドルウェア格納部9504から読み込む。ミドルウェア実行部9503は、ミドルウェア読込部9502が読み込んだミドルウェアを実行する。以上の構成によって、第1～第nのミドルウェアは切り替えて実行される。これによって、図95に示す装置は、複数のミドルウェアを1つのデジタルテレビ上に共存させることを可能としている。

発明の開示

しかし、図95に示した構成では、ミドルウェアの追加や変更への対応が困難であった。以下、詳細を説明する。

例えば、図95に示す装置に新たなミドルウェアを追加

する場合を考える。ここで、一般的に、個々のフォーマットエンジンには独自の状態が規定されている。従って、フォーマットエンジンを管理する側（ミドルウェア読込部 9502 やミドルウェア実行部 9503）は、フォーマットエンジン毎に異なる状態を把握している必要がある。すなわち、新たなミドルウェアが追加される場合、そのミドルウェアにおいて規定されている状態を理解することができるよう、ミドルウェア読込部 9502 やミドルウェア実行部 9503 を変更しなければならない。以上より、新たなミドルウェアを追加する場合には、新たなミドルウェアをミドルウェア格納部 9504 に追加することに加えて、新たなミドルウェアに対応するようミドルウェア読込部 9502 やミドルウェア実行部 9503 を変更しなければならない。従って、図 95 に示すミドルウェア管理装置の設計の際にミドルウェアの追加や変更があると、開発者はミドルウェア読込部 9502 やミドルウェア実行部 9503 についても変更作業を行わなければならない。この変更作業は開発者にとって大きな負担になっていた。

それ故、本発明は、フォーマットエンジンの追加や変更に容易に対応することができる情報処理装置を提供することを目的とする。

上記目的を達成するために、本発明は、以下の構成を採用した。すなわち、本発明は、異なるフォーマットで記述されたデータをそれぞれ実行するためのフォーマットエンジンを格納している情報処理装置である。当該情報処理装置は、フォーマットエンジンの動作状態を全てのフォーマ

ットエンジンに共通の表現によって規定した共通状態を予め定義しておき、各フォーマットエンジンの動作を管理するフォーマットエンジン管理手段と、各フォーマットエンジンに対応して設けられ、共通状態と、フォーマットエンジンの動作状態を各フォーマットエンジン毎に異なる表現によって規定した動作状態である個別状態との対応を予め定義しておき、任意の個別状態となるようにフォーマットエンジンの動作を制御する動作制御手段とを備えている。フォーマットエンジン管理手段は、あるフォーマットエンジンを所定の共通状態に変化させる場合、当該所定の共通状態を示す共通状態情報を含むメッセージを、当該フォーマットエンジンに対応して設けられた動作制御手段へ送信する。各動作制御手段は、フォーマットエンジン管理手段からメッセージが送信されてきた場合、当該メッセージに含まれる共通状態情報により示される共通状態に対応する個別状態となるように、当該フォーマットエンジンを制御する。

また、情報処理装置は、各フォーマットエンジンに対応して設けられ、フォーマットエンジンの個別状態と、当該個別状態に対応する共通状態との組によって構成されるテーブル格納するテーブル格納手段をさらに備えていてよい。このとき、各動作制御手段は、テーブルを参照することによって共通状態から個別状態を決定する。

また、情報処理装置は、各フォーマットエンジンに対応して設けられ、フォーマットエンジンの個別状態を取得し、取得した個別状態に対応する共通状態を示す共通状態情

報をフォーマットエンジン管理手段へ送信する個別状態取得手段をさらに備えていてもよい。このとき、フォーマットエンジン管理手段は、個別状態取得手段から出力された共通状態情報により示される共通状態に基づいて、各フォーマットエンジンの動作を管理する。

また、情報処理装置は、フォーマットエンジンが実行中に利用するリソースであって、複数のフォーマットエンジンが同時に利用することが不可能なリソースである極小リソースをさらに備えていてもよい。このとき、個別状態取得手段は、フォーマットエンジンから取得した個別状態が極小リソースを利用している動作状態を示す場合、当該フォーマットエンジンの共通状態情報として、所定の状態を示す共通状態情報をフォーマットエンジン管理手段へ出力する。また、個別状態取得手段は、フォーマットエンジンから取得した個別状態が極小リソースを利用していない動作状態を示す場合、当該フォーマットエンジンの共通状態情報として、所定の状態以外の状態を示す共通状態情報をフォーマットエンジン管理手段へ出力する。フォーマットエンジン管理手段は、共通状態情報が所定の状態を示すフォーマットエンジンが1つのみになるように、各フォーマットエンジンの動作を管理する。

また、フォーマットエンジン管理手段は、起動受付手段と、共通状態取得手段と、動作停止手段と、起動手段とを含んでいてもよい。起動受付手段は、フォーマットエンジンを起動するための起動要求を受け付ける。共通状態取得手段は、起動受付手段が起動要求を受け付けたことに応じ

て、状態取得手段から各フォーマットエンジンの共通状態情報を取得する。動作停止手段は、共通状態取得手段によって取得された共通状態情報が所定の状態を示すフォーマットエンジンがある場合、当該フォーマットエンジンの動作を停止させるメッセージを当該フォーマットエンジンに対応して設けられた動作制御手段へ送信する。起動手段は、動作停止手段によってフォーマットエンジンの動作が停止された後、起動要求に対応するフォーマットエンジンを起動させるメッセージを当該フォーマットエンジンに対応して設けられた動作制御手段へ送信する。

また、情報処理装置は、極小リソースと、リソース制御手段と、優先度情報格納手段とをさらに備えていてもよい。極小リソースは、フォーマットエンジンが実行中に利用するリソースであって、複数のフォーマットエンジンが同時に利用することが不可能なリソースである。リソース制御手段は、フォーマットエンジンの要求に応じてフォーマットエンジンに対してリソースの利用を許可する。優先度情報格納手段は、極小リソースを利用する場合における各フォーマットエンジン間の優先度を示す優先度情報を格納する。許可決定手段は、極小リソースを利用する要求が複数のフォーマットエンジン間で重複する場合、優先度情報に基づいて、当該極小リソースの利用を許可すべきフォーマットエンジンを決定する。このとき、リソース制御手段は、極小リソースを利用する要求が複数のフォーマットエンジン間で重複する場合、許可決定手段によって決定されたフォーマットエンジンのみに当該極小リソースの利用を

許可し、極小リソースを利用する要求が複数のフォーマットエンジン間で重複しない場合、当該要求を行ったフォーマットエンジンに当該極小リソースの利用を許可する。

また、極小リソースは複数設けられてもよい。このとき、リソース制御手段は極小リソースに対応して複数設けられる。

また、本発明は、異なるフォーマットで記述されたデータをそれぞれ実行するためのフォーマットエンジンを格納している情報処理装置のコンピュータで実行可能なプログラムの形態で提供されてもよい。

本発明によれば、フォーマットエンジン管理手段は各フォーマットエンジンの動作状態を共通状態で管理するので、フォーマットエンジンが追加または変更された場合であっても、フォーマットエンジン管理手段の構成を大幅に変更する必要がない。従って、フォーマットエンジンの追加や変更に容易に対応することができる。さらに、フォーマットエンジン管理部から動作制御部への通信は、共通状態情報を含むメッセージによって行われる。従って、フォーマットエンジン管理部は、フォーマットエンジンの違いを意識せずに各フォーマットエンジンの動作状態を管理できるので、複数のフォーマットエンジンの動作を同時に制御することが容易になる。

また、情報処理装置がテーブル格納手段をさらに備えることによって、個別状態と共通状態との相互の変換を容易に行うことができる。

また、さらに、情報処理装置が個別状態取得手段をさら

に備えることによって、フォーマットエンジン管理手段は、各フォーマットエンジンの動作状態を容易に取得することができる。

また、さらに、フォーマットエンジン管理手段が、共通状態情報が所定の状態を示すフォーマットエンジンが1つのみになるように、各フォーマットエンジンの動作を管理することによって、次の効果を得ることができる。すなわち、フォーマットエンジン管理手段は、極小リソースの利用が競合しないように、フォーマットエンジンの動作を管理することができる。換言すれば、フォーマットエンジン管理手段は、極小リソースの利用が競合しない範囲で、複数のフォーマットエンジンを同時に実行することも可能である。

また、さらに、フォーマットエンジン管理手段が、起動受付手段と、共通状態取得手段と、動作停止手段と、起動手段とを含んでいる場合には、次の効果を得ることができる。すなわち、フォーマットエンジン管理手段は、起動要求に対応するフォーマットエンジンを確実に起動させることができるとともに、極小リソースの利用が競合しない範囲で、起動中のフォーマットエンジンの実行を継続することができる。

また、さらに、情報処理装置が、極小リソースと、リソース制御手段と、優先度情報格納手段とをさらに備えている場合には、以下の効果を得ることができる。すなわち、フォーマットエンジン管理手段は、極小リソースの利用が競合しないように、フォーマットエンジンの動作を管理す

ることができる。換言すれば、フォーマットエンジン管理手段は、極小リソースの利用が競合しない範囲で、複数のフォーマットエンジンを同時に実行することも可能である。さらに、この場合、互いに異なる複数のフォーマットエンジンは、それぞれ、互いに異なる複数の極小リソースを同時に利用することも可能である。

また、極小リソースが複数設けられている場合には、各フォーマットエンジンは、それぞれ別個の極小リソースを同時に利用することができる。

図面の簡単な説明

図1は、本実施の形態に係る情報処理装置の一例であるデジタルテレビのハードウェア構成を表すブロック図である。

図2は、図1に示すデジタルテレビ100の外観を示す図である。

図3は、入力部111を構成するフロントパネルの一例を示す図である。

図4は、第1の動作例を行う場合における、図1に示すデジタルテレビの機能的な構成を示すブロック図である。

図5は、テーブル格納部44が格納するテーブルの一例を示す図である。

図6は、第2の動作例を行う場合における、図1に示すデジタルテレビの機能的な構成を示すブロック図である。

図7は、情報処理装置の機能的な構成をより具体的に示す図である。

図 8 は、ディスプレイ 107 に表示されるフォーマットエンジンの一覧の例を示す図である。

図 9 は、ディスプレイ 107 に表示されるフォーマットエンジンの一覧の例を示す図である。

図 10 は、ディスプレイ 107 に表示されるフォーマットエンジンの一覧の例を示す図である。

図 11 は、ディスプレイ 107 に表示される、実行可能なアプリケーションの一覧の例を示す図である。

図 12 は、ディスプレイ 107 に表示される、実行可能なアプリケーションの一覧の例を示す図である。

図 13 は、ディスプレイ 107 に表示される、実行可能なアプリケーションの一覧の他の例を示す図である。

図 14 は、ディスプレイ 107 に表示される、実行可能なアプリケーションの一覧の他の例を示す図である。

図 15 は、ディスプレイ 107 に表示される、実行可能な H T M L データの一覧の例を示す図である。

図 16 は、ディスプレイ 107 に表示される、実行可能な H T M L データの一覧の例を示す図である。

図 17 は、メッセージフォーマットの一例を示す図である。

図 18 は、サブプログラム I D の表の一例を示す図である。

図 19 は、Message I D フィールド 173 に格納されるメッセージ I D の表の一例を示す図である。

図 20 は、フォーマットエンジン状態応答メッセージにおけるデータフィールド 175 のフォーマットの一例を示す図である。

す図である。

図21は、フォーマットエンジンの動作状態と動作状態IDとの対応を示す表の一例を示す図である。

図22は、アプリケーション・データ一覧応答メッセージにおけるデータフィールド175のフォーマットの一例を示す図である。

図23は、「アプリケーション・データ実行」「アプリケーション・データ停止」「アプリケーション・データ時停止」のメッセージにおけるデータフィールド175のフォーマットの一例を示す図である。

図24は、「フォーマットエンジン状態要求」メッセージの具体例を示す図である。

図25は、「フォーマットエンジン状態要求」メッセージの具体例を示す図である。

図26は、「フォーマットエンジン状態要求」メッセージの具体例を示す図である。

図27は、「フォーマットエンジン取得応答」メッセージの具体例を示す図である。

図28は、「フォーマットエンジン取得応答」メッセージの具体例を示す図である。

図29は、「フォーマットエンジン取得応答」メッセージの具体例を示す図である。

図30は、「アプリケーション・データ一覧要求」メッセージの具体例を示す図である。

図31は、「アプリケーション・データ取得応答」メッセージの具体例を示す図である。

図 3 2 は、状態管理部 7 3 2 が保持している情報の一例を示す図である。

図 3 3 は、状態管理部 7 3 2 が保持している情報の一例を示す図である。

図 3 4 は、状態管理部 7 3 2 が保持している情報の一例を示す図である。

図 3 5 は、「フォーマットエンジン状態変化」メッセージの一例を示す図である。

図 3 6 は、状態管理部 7 3 2 が送信するメッセージの一例を示す図である。

図 3 7 は、メーラー 7 6 0 が送信するメッセージの一例を示す図である。

図 3 8 は、図 3 2 の変化後の状態を示す図である。

図 3 9 は、リソース管理部 7 3 3 の内部構成を示すプロック図である。

図 4 0 は、プロセス記憶部 3 9 0 1 において記憶されている情報の一例を示す図である。

図 4 1 は、優先度記憶部 3 9 0 2 に記憶されている情報の一例を示す図である。

図 4 2 は、最新起動記憶部 3 9 0 3 が記憶している情報の一例を示す図である。

図 4 3 は、リソース ID 記憶部 3 9 0 4 が記憶している情報の一例を示す図である。

図 4 4 は、リソース ID の定義をプログラミング言語である C 言語で記述した例を示す図である。

図 4 5 は、Java ミドルウェア 7 4 0 の構成を示すブ

ロック図である。

図46は、AITの主要部を表した模式図である。

図47は、Javaアプリケーションの4つの動作状態および動作状態間の遷移を表す状態遷移図である。

図48は、AITの主要部を表した模式図である。

図49は、Javaミドルウェア740が実行可能な全てのJavaアプリケーションの動作状態と、Javaミドルウェア740の動作状態との対応を示す変換表の一例を示す図である。

図50は、各Javaアプリケーションの動作状態を共通状態に変換するための変換表の一例を示す図である。

図51は、第1の変換部741が送信するメッセージの具体例を示す図である。

図52は、HTMLブラウザ750の構成を示すプロック図である。

図53は、DVB-HHTMLデータの一例を示す図である。

図54は、VB-HHTMLデータのレイアウトを決めるスタイルシートの一例を示す図である。

図55は、DOMツリーの例を示す図である。

図56は、図55に示すDOMツリーを描画した際のディスプレイ107を示す図である。

図57は、DVB-HHTMLデータの例を示す図である。

図58は、DVB-HHTMLデータの例を示す図である。

図 5 9 は、図 5 8 に示す D V B - H T M L データに対してパーザー 5 2 0 1 が構築した D O M ツリーの一部を示す図である。

図 6 0 は、レイアウター 5 2 0 2 および描画部 5 2 0 3 によって表示されたディスプレイ表示の例を示す図である。

図 6 1 は、D O M ツリーの変更結果を示す図である。

図 6 2 は、ディスプレイ表示の変更結果の例を示す図である。

図 6 3 は、A I T の主要部を表した模式図である。

図 6 4 は、D V B - H T M L データの 5 つの状態および状態間の遷移を表す状態遷移図である。

図 6 5 は、A I T の主要部を表した模式図である。

図 6 6 は、D V B - H T M L データ内に他の D V B - H T M L データへのリンクが定義されている例を示す図である。

図 6 7 は、図 6 6 に示す D V B - H T M L データの表示例を示す図である。

図 6 8 は、D V B - H T M L データの表示例を示す図である。

図 6 9 は、H T M L ブラウザ 7 5 0 が表示可能な全ての H T M L データの状態と、H T M L ブラウザ 7 5 0 の動作状態との対応を示す変換表の一例を示す図である。

図 7 0 は、各 H T M L データの状態を共通状態に変換するための変換表の一例を示す図である。

図 7 1 は、第 2 の変換部 7 5 1 が送信するメッセージの

具体例を示す図である。

図72は、メーラー760の状態と、結合部730が規定している共通状態との対応を示す変換表の一例を示す図である。

図73は、第3の変換部761が生成する「アプリケーション・データー覧応答」メッセージの具体例を示す図である。

図74は、第1の動作例の動作を行う場合におけるデジタルテレビにおける処理の流れを示すフローチャートである。

図75は、通信部731が行うメッセージ送信の処理の流れを示すフローチャートである。

図76は、第1の変換部741が「フォーマットエンジン状態要求」メッセージを受け取ったときの処理の流れを示すフローチャートである。

図77は、第3の変換部771が「フォーマットエンジン状態要求」メッセージを受け取ったときの処理の流れを示すフローチャートである。

図78は、第1の変換部741が、「フォーマットエンジン停止」メッセージを受け取った際の処理の流れを表すフローチャートである。

図79は、第2の変換部751が、「フォーマットエンジン停止」メッセージを受け取った際の処理の流れを示すフローチャートである。

図80は、第3の変換部761が、「フォーマットエンジン停止」メッセージを受け取った際の処理の流れを示す

フロー チャートである。

図 8 1 は、第 1 の変換部 741 が、「フォーマットエンジン実行」メッセージを受け取った際の処理の流れを示すフロー チャートである。

図 8 2 は、第 1 の変換部 741 が、「フォーマットエンジン一時停止」メッセージを受け取った際の処理の流れを表すフロー チャートである。

図 8 3 は、第 2 の動作例の動作を行う場合におけるデジタルテレビにおける処理の流れを示すフロー チャートである。

図 8 4 は、リソース剥奪通知を受け取った変換部における処理の流れを示すフロー チャートである。

図 8 5 は、第 1 の変換部 741 がアプリケーションマネージャ 4503 から Java アプリケーションの状態変化通知を受けた際の処理の流れを示すフロー チャートである。

図 8 6 は、第 1 の変換部 741 が「アプリケーション・データ一覧要求」メッセージを受け取った際の処理の流れを示すフロー チャートである。

図 8 7 は、第 1 の変換部 741 が、「アプリケーション・データ実行」メッセージを受け取った際の処理の流れを示すフロー チャートである。

図 8 8 は、第 1 の変換部 741 が、「アプリケーション・データ停止」メッセージを受け取った際の処理の流れを示すフロー チャートである。

図 8 9 は、第 1 の変換部 741 が、「アプリケーション

・データ一時停止」メッセージを受け取った際の処理の流れを示すフローチャートである。

図90は、第3の変換部761がメーラー760が終了したことの通知をメーラー760から受けた際の処理の流れを示すフローチャートである。

図91は、結合部730が「フォーマットエンジン状態変化」メッセージを受け取った場合の結合部730の状態管理部732の処理の流れを示すフローチャートである。

図92は、結合部730が「アプリケーション・データ一覧変化」メッセージを受け取った場合の結合部730の状態管理部732の処理の流れを示すフローチャートである。

図93は、フォーマットエンジンを実行しても、他のフォーマットエンジンに大きな影響を与えない状況の組み合わせを示す図である。

図94は、従来のデジタル放送受信機の構成図の一例である。

図95は、従来のデジタルテレビにおいて用いられるミドルウェア切替装置の構成図である。

発明を実施するための最良の形態

以下、本発明に係る情報処理装置の実施の形態を、図面を参照しながら説明する。なお、以下の実施の形態では、情報処理装置の一例としてデジタルテレビを用いて説明するが、本発明は携帯電話や携帯情報端末等においても適用可能である。

(ハードウェア構成)

まず、図1～図3を参照して、情報処理装置のハードウェア構成について説明する。図1は、本実施の形態に係る情報処理装置の一例であるデジタルテレビのハードウェア構成を表すブロック図である。図1において、デジタルテレビ100は、チューナ101、デスクランプ102、T Sデコーダ103、オーディオデコーダ104、スピーカ105、ビデオデコーダ106、ディスプレイ107、2次記憶部108、1次記憶部109、R O M110、入力部111、ネットワークインターフェース112、およびC P U113を備えている。

図2は、図1に示すデジタルテレビ100の外観を示す図である。図1に示すデジタルテレビ100は、例えば図2に示すような薄型テレビとして実現される。図2に示すように、デジタルテレビ100は、図1に示す構成に加え、筐体201および信号入力端子204をさらに備えている。なお、ディスプレイ202は、図1に示すディスプレイ107に相当する。また、フロントパネル部203は、図1に示す入力部111に相当する。フロントパネル部203は、複数のボタンで構成される。なお、フロントパネル部203の詳細な構成は後述する図3に示されている。信号入力端子204は、地上波放送の放送局、衛星のアンテナ、あるいはケーブルテレビの局システムからの放送信号をデジタルテレビ100に入力するための端子である。信号入力端子204は、図1のチューナ101と接続されている。なお、デジタルテレビ100は、インターネット

に接続するためのネットワーク端子を有している。ネットワーク端子は、ネットワークインターフェース 112 と接続されている。

次に、デジタルテレビ 100 の基本的な動作を説明する。チューナ 101 は、信号入力端子 204 を介して送信されてきた放送信号を入力する。チューナ 101 は、C P U 113 によって指定された周波数を示すチューニング情報に従って、入力した放送信号を復調する。復調された放送信号は、デスクランプ 102 に引き渡される。デスクランプ 102 は、復調された放送信号を復号する。復号に必要な鍵は C P U 113 から与えられる。復号によって得られる M P E G 2 トランSPORTストリームは、T S デコーダ 103 へ出力される。

T S デコーダ 103 は、デスクランプ 102 から受け取った M P E G 2 トランSPORTストリームに対してフィルタリングを行い、必要なデータをオーディオデコーダ 104、ビデオデコーダ 106 あるいは C P U 113 に引き渡す。M P E G 2 トランSPORTストリームの詳細は M P E G 規格書 “ I S O / I E C 1 3 8 1 8 - 1 ” に記載されているので詳細な記述は省略する。M P E G 2 トランSPORTストリームは、複数の固定長パケットで構成され、各パケットには映像、音声、字幕データ、およびアプリケーション等が格納されている。また、各パケットにはパケット I D が与えられている。T S デコーダ 103 は、C P U 113 によってパケット I D とそのパケット I D を有するパケットの出力先の組を指示される。パケットの出力先は

「オーディオデコーダ 104」、「ビデオデコーダ 106」、および「C P U 113」のいずれかである。例えば、C P U 113が、パケットI Dが「123」であるパケットの出力先が「ビデオデコーダ 106」である旨の指示をT S デコーダ 103へ出力した場合、T S デコーダ 103は、M P E G 2 トランスポートストリームからパケットI Dが「123」であるパケットを抽出し、そのパケットをビデオデコーダ 106に引き渡す。なお、T S デコーダは、複数のフィルタリング処理を変更して実行することができる。

オーディオデコーダ 104は、T S デコーダ 103から引き渡された各パケットに埋め込まれているオーディオデータを連結し、デジタルーアナログ変換を行いスピーカ 105へ出力する。スピーカ 105は、オーディオデコーダ 104から出力された信号を音声出力する。ビデオデコーダ 106は、T S デコーダ 103から引き渡された各パケットに埋め込まれているビデオデータを連結し、デジタルーアナログ変換を行うことによって得られるビデオ信号をディスプレイ 107へ出力する。ディスプレイ 107は、典型的にはブラウン管や液晶表示装置等で構成され、ビデオデコーダ 106から出力されたビデオ信号を出力する。さらに、ディスプレイ 107は、C P U 113から指示されたメッセージ等を表示することもある。

2次記憶部 108は、典型的にはフラッシュメモリやハードディスク等で構成され、C P U 113から指示されたデータやプログラムを保存する。保存されているデータや

プログラムは C P U 1 1 3 に参照される。保存されているデータやプログラムは、デジタルテレビ 1 0 0 の電源が切断された状態でも保存しつづける。1 次記憶部 1 0 9 は、典型的には R A M 等で構成され、C P U 1 1 3 から指示されたデータやプログラムを一次的に保存する。保存されているデータやプログラムは C P U 1 1 3 に参照される。保存されているデータやプログラムは、デジタルテレビ 1 0 0 の電源が切断された際に、抹消される。R O M 1 1 0 は、書き換え不可能なメモリデバイスであり、典型的には R O M や C D - R O M 、 D V D 等で構成される。R O M 1 1 0 には、C P U 1 1 3 によって実行されるプログラムが格納されている。

入力部 1 1 1 は、典型的にはフロントパネルやリモコンで構成され、ユーザからの入力を受け付ける。図 3 は、入力部 1 1 1 を構成するフロントパネルの一例を示す図である。図 3 において、フロントパネル 3 0 0 は 7 つのボタン、すなわち、上カーソルボタン 3 0 1 、下カーソルボタン 3 0 2 、左カーソルボタン 3 0 3 、右カーソルボタン 3 0 4 、OK ボタン 3 0 5 、取消ボタン 3 0 6 、M E N U ボタン 3 0 7 、C + ボタン 3 0 8 、および C - ボタン 3 0 9 を備えている。ユーザが各ボタンを押下すると、押下されたボタンを示す識別子が C P U 1 1 3 に通知される。

ネットワークインターフェース 1 1 2 は、典型的にはモデムやイーサネットコネクター等で構成され、インターネットやイントラネット等のネットワークと接続する機能をデジタルテレビ 1 0 0 に提供するものである。ネットワー

クインターフェース 112 は、CPU113 からの指示に従い、ネットワーク上に接続された他の情報機器とのデータの送受信を行う。例えばネットワーククインターフェース 112 がモデムによって構成されている場合、ネットワーククインターフェース 112 は、CPU113 から接続先の電話番号や通信パラメータ情報を受け取り、ネットワークとの通信経路を確立する。

CPU113 は、ROM110 が記憶するプログラムを実行する。CPU113 は、プログラムの指示に従い、チューナ 101、デスクランプ 102、TS デコーダ 103、ディスプレイ 107、2 次記憶部 108、1 次記憶部 109、および ROM110 の動作を制御する。CPU113 の詳細な動作については後述する。

(動作概要)

次に、情報処理装置の詳細な構成および動作を説明する前に、図 4～図 6 を参照して、本発明の目的を達成するための情報処理装置の動作の概要を説明する。ここでは、情報処理装置の動作例として、第 1 および第 2 の動作例という 2 つの動作例について説明する。

(第 1 の動作例)

まず、図 4 および図 5 を用いて第 1 の動作例の概要を説明する。図 4 は、第 1 の動作例を行う場合における、図 1 に示すデジタルテレビの機能的な構成を示すブロック図である。図 4 において、デジタルテレビは、フォーマットエンジン管理部 41 と、第 1 の変換部 42 と、第 2 の変換部 43 と、第 1 のテーブル格納部 44 と、第 2 のテーブル格

納部 4 9 と、第 1 のフォーマットエンジン 4 5 と、第 2 のフォーマットエンジン 4 6 と、リソース制御部 4 7 と、リソース 4 8 とを備えている。図 4 において、リソース 4 8 以外の各構成要素は、図 1 に示す C P U 1 1 3 が R O M 1 1 0 に格納されたプログラムを 1 次記憶部 1 0 9 および 2 次記憶部 1 0 8 を用いて実行することによって実現される。また、リソース 4 8 は、フォーマットエンジンによって利用されるハードウェアである。リソース 4 8 は、例えば、メモリ（1 次記憶部 1 0 9 および 2 次記憶部 1 0 8 ）や、チューナ 1 0 1 、 T S デコーダ 1 0 3 、ネットワークインターフェース 1 1 2 等である。なお、図 4 では、説明を容易にする目的で、デジタルテレビ 1 0 0 が有するフォーマットエンジンの数を 2 つとして説明するが、フォーマットエンジンの数は 2 つ以上であってもよい。

ここで、各フォーマットエンジン 4 5 および 4 6 は、複数種類の動作状態を取り得る。すなわち、各フォーマットエンジン 4 5 および 4 6 は、「実行されている状態」や、「実行されていない状態」等の状態を取り得る。以下、情報処理装置の動作概要を説明する図 4 ～ 図 6 においては、第 1 のフォーマットエンジン 4 5 は、「動作中」、「停止中」、および「一時停止中」と表現される 3 種類の動作状態を取り得るものとする。また、第 2 のフォーマットエンジン 4 6 は、「起動中」および「起動中でない」という 2 種類の動作状態を取り得るものとする。ここで、各フォーマットエンジンにおいて個別に規定された動作状態を、個別状態と呼ぶ。つまり、第 1 のフォーマットエンジン 4 5

においては、「動作中」、「停止中」、および「一時停止中」と表現される3種類の動作状態を含む個別状態が規定されている。また、第2のフォーマットエンジン46においては、「起動中」および「起動中でない」という2種類の動作状態を含む個別状態が規定されている。なお、一般的に、各フォーマットエンジンが取り得る動作状態の種類は、フォーマットエンジンによって様々である。

フォーマットエンジン管理部41は、各フォーマットエンジン45および46の動作を管理する。具体的には、各フォーマットエンジン45および46に対して、起動や停止等の指示を行う。ここで、フォーマットエンジン管理部41においては、各フォーマットエンジンが取り得る動作状態は、共通状態として予め規定されている。共通状態とは、全てのフォーマットエンジンに共通の表現によってフォーマットエンジンの動作状態を規定した概念である。図4においては、フォーマットエンジン管理部41においては、「実行中」および「終了」と表現される2種類の動作状態を含む共通状態が規定されるものとする。このように、共通状態は、各フォーマットエンジン45および46において規定されている動作状態（すなわち、個別状態）とは異なる表現であってもよい。また、共通状態に含まれる動作状態の種類は2種類に限らない。共通状態に含まれる動作状態の種類は、フォーマットエンジン管理部41が各フォーマットエンジンを管理するために必要となる数の動作状態の種類を定義しておけばよい。

また、フォーマットエンジン管理部41は、例えばユ

ザからの入力に従って、フォーマットエンジンに対して動作を指示する。このとき、フォーマットエンジン管理部41は、共通状態情報を用いて指示を行う。共通状態情報とは、フォーマットエンジンの動作状態を共通状態によって表現した情報である。図4において、例えば、フォーマットエンジン管理部41が第1のフォーマットエンジン45の動作状態が所定の動作状態となるように指示を行う場合、フォーマットエンジン管理部41は、当該所定の動作状態を示す共通状態情報を第1の変換部42へ出力する。これと同様に、フォーマットエンジン管理部41は、第2のフォーマットエンジン46に対して動作の指示を行う場合、第2の変換部43へ共通状態情報を出力する。

また、フォーマットエンジン管理部41は、各変換部42および43から出力されてくる共通状態情報を受け取ることによって、各フォーマットエンジン45および46の動作状態を共通状態の表現によって把握する。このように把握された共通状態を用いて、フォーマットエンジン管理部41は、各フォーマットエンジン45および46の動作を管理することができる。

第1の変換部42は、フォーマットエンジン管理部41から出力されてくる共通状態情報に基づいて第1のフォーマットエンジン45の動作を制御する。すなわち、第1の変換部42は、フォーマットエンジン管理部41において規定される共通状態と、第1のフォーマットエンジン45において規定される個別状態との対応を予め規定しておく。フォーマットエンジン管理部41から共通状態情報が出

力されてくると、第1の変換部42は、当該対応に基づいて、共通状態情報により示される共通状態に対応する個別状態を特定する。さらに、第1の変換部42は、特定した個別状態となるように第1のフォーマットエンジン45の動作を制御する。

また、第1の変換部42は、第1のフォーマットエンジン45の動作状態（個別状態で表現されている）を取得し、取得した動作状態を共通状態によって表現される形でフォーマットエンジン管理部41へ通知する。つまり、第1の変換部42は、第1のフォーマットエンジン45から取得した動作状態を示す共通状態情報をフォーマットエンジン管理部41へ出力する。共通状態情報をフォーマットエンジン管理部41へ出力する場合も、上記共通状態と個別状態との対応が用いられる。すなわち、第1のフォーマットエンジン45から個別状態を取得すると、第1の変換部42は、上記対応に基づいて、取得した個別状態に対応する共通状態を特定する。さらに、第1の変換部42は、特定した共通状態を示す共通状態情報をフォーマットエンジン管理部41へ出力する。

第2の変換部43は、第1の変換部42と同様の処理を行う。すなわち、第2の変換部43は、フォーマットエンジン管理部41から出力されてくる共通状態情報に基づいて第2のフォーマットエンジン46の動作を制御する。また、第2の変換部43は、第2のフォーマットエンジン46の動作状態（個別状態で表現されている）を取得し、取得した動作状態を共通状態によって表現される形でフォー

マットエンジン管理部 4 1 へ通知する。これらの動作の詳細は、第 1 の変換部 4 2 における動作と同様の方法で行われる。

第 1 のテーブル格納部 4 4 は、フォーマットエンジン管理部 4 1 において規定される共通状態と、第 1 のフォーマットエンジン 4 5 において規定される個別状態とを対応付けたテーブルを格納する。また、第 2 のテーブル格納部 4 4 は、フォーマットエンジン管理部 4 1 において規定される共通状態と、第 2 のフォーマットエンジン 4 6 において規定される個別状態とを対応付けたテーブルを格納する。

図 5 は、各テーブル格納部 4 4 および 4 9 が格納するテーブルの一例を示す図である。図 5 (a) は、第 1 のテーブル格納部 4 4 が格納するテーブルの一例を示す図であり、図 5 (b) は、第 2 のテーブル格納部 4 9 が格納するテーブルの一例を示す図である。図 5 (a) においては、「実行中」という共通状態に対して、「動作中」という個別状態が対応付けられる。また、「終了」という共通状態に対して、「停止中」および「一時停止中」という個別状態が対応付けられる。図 5 (b) においては、「実行中」という共通状態に対して、「起動中」という個別状態が対応付けられている。また、「終了」という共通状態に対して、「起動中でない」という個別状態が対応付けられている。各変換部 4 2 および 4 3 は、図 5 に示すテーブルを参照して、フォーマットエンジン管理部 4 1 から出力されてくる共通状態情報により示される共通状態に対応する個別状態を特定する。また、各変換部 4 2 および 4 3 は、図 5 に示

すテーブルを参照して、各フォーマットエンジン 4 5 および 4 6 から取得した個別状態に対応する共通状態を特定する。

例えば、第 1 の変換部 4 5 に「実行中」を示す共通状態情報が出力されてきた場合、第 1 の変換部 4 5 は、図 5 (a) に示すテーブルを参照して、「動作中」という個別状態を特定する。そして、第 1 の変換部 4 5 は、特定した「動作中」という動作状態となるように第 1 のフォーマットエンジン 4 5 を制御する。また、例えば、第 2 の変換部 4 3 が第 2 のフォーマットエンジン 4 6 から「起動中でない」を示す個別状態を取得した場合、第 2 の変換部 4 3 は、上記図 5 (b) に示すテーブルを参照して、「終了」という共通状態を特定する。そして、第 2 の変換部 4 3 は、特定した「終了」を示す共通状態情報をフォーマットエンジン管理部 4 1 へ出力する。

また、図 4において、リソース制御部 4 7 は、各フォーマットエンジン 4 5 および 4 6 の要求に応じて、各フォーマットエンジン 4 5 および 4 6 に対してリソース 4 8 の利用を許可する。

なお、以上の動作は、後述する第 2 の動作例においても同様である。

ここで、リソース 4 8 が極小リソースである場合を考える。極小リソースとは、フォーマットエンジンが実行中に利用するリソースであって、複数のフォーマットエンジンが同時に利用することが不可能なリソースである。例えば、チューナ 101 や、TS デコーダ 103 や、ネットワー

クインターフェース 112 は極小リソースである。極小リソースは、デジタルテレビ 100 が有している数が限られたりソースで、かつ、複数のフォーマットエンジンが同時に使用を試み、競合する可能性のあるリソースのことである。例えば、チューナ 101 は 1 つの周波数帯にしかチューニングできない。複数のフォーマットエンジンが異なる周波数帯にチューニングしたい場合にリソースの競合が発生する。このような極小リソースは複数のフォーマットエンジンが同時に利用することができないので、リソース制御部 47 は、複数のフォーマットエンジンに対して極小リソースの利用を同時に許可しないようにする必要がある。もしリソース制御部 47 が複数のフォーマットエンジンに対して極小リソースの利用を同時に許可すれば、当該フォーマットエンジンの動作が正常に行われない可能性がある。そこで、図 4においては、フォーマットエンジンの動作を正常に行うための方法として、以下の方法をとる。

図 4においては、各テーブル格納部 44 および 49 に格納されているテーブルにおける共通状態と個別状態との対応付けは、次のように行われる。すなわち、フォーマットエンジンが極小リソースを利用する動作状態を表す個別状態に対して、「実行中」という共通状態が対応付けられる。さらに、フォーマットエンジンが極小リソースを利用しない動作状態を表す個別状態に対して、「終了」という共通状態が対応付けられる。なお、図 4においては、第 1 のフォーマットエンジン 45 は、「動作中」という個別状態においてのみ極小リソースを利用し、他の個別状態において

ては極小リソースを利用しないものとする。また、第2のフォーマットエンジン46は、「起動中」という個別状態においてのみ極小リソースを利用し、他の個別状態においては極小リソースを利用しないものとする。このとき、上記テーブルは図5に示すようになる。

上記テーブルの対応付けを以上のように行うとともに、さらに、フォーマットエンジン管理部41は、共通状態が「実行中」であるフォーマットエンジンが1つになるよう、各フォーマットエンジン45および46の動作を管理する。これによって、フォーマットエンジン管理部41は、極小リソースを利用するフォーマットエンジンが1つになるように各フォーマットエンジン45および46の動作を管理することができる。フォーマットエンジン管理部41による具体的な管理方法は種々考えられる。具体的な管理方法については、後述する処理例において述べる。

(第2の動作例)

次に、図6を用いて第2の動作例について説明する。図6は、第2の動作例を行う場合における、図1に示すデジタルテレビの機能的な構成を示すブロック図である。図6において、デジタルテレビは、図4に示す構成に加えて、許可決定部61と、優先度情報格納部62とを備えている。なお、図6において、図4に示す構成要素と同じ構成要素には同じ参照符号を付している。図6において、許可決定部61および優先度情報格納部62は、図1に示すCPU113がROM110に格納されたプログラムを1次記憶部109および2次記憶部108を用いて実行すること

によって実現される。

図4における動作と図6における動作との相違点は、リソース48が極小リソースである場合における、フォーマットエンジンの動作を正常に行うための方法である。その他の動作、例えば、フォーマットエンジン管理部41が共通状態によって各フォーマットエンジン45および46の動作を管理する動作や、各変換部42および43が共通状態と個別状態との相互の変換を行う動作については、図4と同様である。従って、以下では、第2の動作例においてフォーマットエンジンの動作を正常に行うための方法を説明する。

第2の動作例では、各テーブル格納部44および49において、フォーマットエンジンが極小リソースを利用する動作状態を表す個別状態に対して、「実行中」という共通状態が対応付けられる必要はない。さらに、フォーマットエンジン管理部41は、共通状態が「実行中」であるフォーマットエンジンが1つになるように、各フォーマットエンジン45および46の動作を管理する必要はない。第2の動作例では、リソース48を利用する要求がリソース制御部47において重複した場合、リソースを使用可能なフォーマットエンジンが優先度情報に基づいて選択される。以下、詳細を説明する。

優先度情報格納部62は、優先度情報を格納する。優先度情報とは、極小リソースであるリソース48を利用する場合における各フォーマットエンジン間の優先度を示す情報である。図6においては、優先度情報は、第1のフォー

マットエンジン 4 5 と第 2 のフォーマットエンジン 4 6 とのいずれを優先するかを示す情報である。

リソース制御部 4 7 は、リソース 4 8 を利用する要求がリソース制御部 4 7 において重複した場合、すなわち、リソース 4 8 を利用する要求が複数のフォーマットエンジンから同時に行われた場合、要求が重複した旨の通知を許可決定部 6 1 へ出力する。

許可決定部 6 1 は、要求が重複した旨の通知をリソース制御部 4 7 から受け取った場合、優先度情報格納部 6 2 に格納されている優先度情報に基づいて、リソース 4 8 の利用を許可すべきフォーマットエンジンを決定する。さらに、許可決定部 6 1 は、決定したフォーマットエンジンを示す情報をリソース制御部 4 7 へ出力する。リソース制御部 4 7 は、当該情報により示されるフォーマットエンジンにリソース 4 8 の利用を許可する。以上によって、極小リソースを利用するフォーマットエンジンが 1 つになるよう各フォーマットエンジン 4 5 および 4 6 の動作を管理することができる。なお、利用が許可されなかったフォーマットエンジンに対して、利用を許可しない旨の通知がリソース制御部 4 7 またはフォーマットエンジン管理部 4 1 によって行われてもよい。

以上のように、上記第 1 および第 2 動作例によれば、フォーマットエンジン管理部 4 1 は共通状態を用いて各フォーマットエンジン 4 5 および 4 6 の動作を管理する。共通状態は各フォーマットエンジンにおいて規定される個別状態とは別個であり、管理すべきフォーマットエンジンとは

無関係である。従って、仮にフォーマットエンジンの追加や変更があった場合でも、共通状態として規定されている内容を変更する必要はない。つまり、フォーマットエンジン管理部41を変更する必要はない。フォーマットエンジンの追加があった場合には、当該フォーマットエンジンを追加するとともに、それに対応する変換部を追加すればよいだけである。つまり、追加するフォーマットエンジンにおいて規定される個別状態と共通状態との対応を新たに追加すればよいだけである。以上のように、上記第1および第2動作例によれば、仮にフォーマットエンジンの追加や変更があった場合でも、当該追加や変更に容易に対応することができる。

さらに、リソース48が極小リソースである場合、上記第1および第2動作例によれば、極小リソースを複数のフォーマットエンジンが同時に利用しないように各フォーマットエンジンの動作が管理される。従って、極小リソースを利用する複数のフォーマットエンジンが正常に動作しなくなることを防止することができる。

なお、上記第1および第2の動作例においては、リソース48が1つである場合について説明したが、リソースは複数であってもよい。リソースが複数である場合、リソース制御部はリソースに対応して、リソースと同数設けられる必要がある。なお、第1の動作例においてリソースが複数である場合であっても、フォーマットエンジンが極小リソースを利用する動作状態を表す個別状態に対して、「実行中」という共通状態が対応付けられる点は上記第1の動

作例と同様である。また、第1の動作例においてリソースが複数である場合であっても、フォーマットエンジン管理部41が、共通状態が「実行中」であるフォーマットエンジンが1つになるように、各フォーマットエンジン45および46の動作を管理する点は上記第1の動作例と同様である。

また、第2の動作例においてリソースが複数である場合、極小リソースに対応して設けられるリソース制御部は、上記第2の動作例と同様の動作を行う。これによって、極小リソースが複数である場合、各極小リソースが重複して利用されることを防止することができる。さらに、第2の動作例では、互いに異なる複数のフォーマットエンジンは、それぞれ、互いに異なる複数の極小リソースを同時に利用することが可能である。つまり、あるフォーマットエンジンAが極小リソースAを利用している場合に、別のフォーマットエンジンBは極小リソースBを利用することができる。なお、第1の動作例では、極小リソースが複数である場合、複数の極小リソースを同時に利用することは不可能であった。これに対して第2の動作例では、複数の極小リソースを同時に利用することができる点で、第1の動作例よりも効率よくフォーマットエンジンを実行することができると言える。

なお、上記第1および第2の動作例においては、各変換部45および46は、フォーマットエンジンと別個に構成されるものとしたが、フォーマットエンジンと一体的に構成されるものであってもよい。換言すれば、フォーマット

エンジンは、変換部の機能を有するものであってもよい。

以上で、第1および第2の動作例の概要の説明を終了する。

(具体的な構成例)

次に、図7～図73を用いて、本発明に係る情報処理装置の具体的な構成を説明する。なお、以下で説明する構成は、上述した第1および第2の動作例の双方を実現することが可能な構成である。従って、以下で説明する構成には、第1の動作例のみを実行する場合には不要である構成や、第2の動作例のみを実行する場合には不要である構成が含まれるが、そのような構成については説明中において適宜述べる。

図7は、情報処理装置の機能的な構成をより具体的に示す図である。図7に示す各構成要素は、図1に示すCPU113がROM110に格納されたプログラムを1次記憶部109および2次記憶部108を用いて実行することによって実現される。換言すれば、図7は、ROM110に記憶され、CPU113に実行されるプログラムの構成を示す図である。

図7において、情報処理装置は、OS710と、ナビゲータ720と、結合部730と、Javaミドルウェア740と、HTMLブラウザ450と、メーラー460と、第1の変換部741と、第2変換部451と、第3の変換部761とを備えている。これらは、ROM110に格納されているプログラムのサブプログラムである。なお、図4に示すフォーマットエンジン管理部41は、ナビゲータ

720 および結合部 730 の機能の一部として実現される。

図 7 に示すサブプログラムの内、Java ミドルウェア 740、HTML ブラウザ 750、およびメーラー 760 は、所定のフォーマットで記述されたアプリケーションの実行や、所定のフォーマットで記述されたデータの表示等を行うフォーマットエンジンである。以下に説明する具体的な構成例では、これら 3 種類のフォーマットエンジンを例として取り上げたが、フォーマットエンジンは、ワードプロセッサプログラムやスプレッドシートプログラム等であってもよい。また、情報処理装置が備えるフォーマットエンジンの数は 3 つに限るものではなく、1 つでも 2 つでもよく、4 つ以上でもよい。ただし、フォーマットエンジンの数と同数の変換部を用意する必要がある。

OS 710 は、デジタルテレビ 100 の電源が投入されると、CPU 113 が起動するサブプログラムである。OS 710 は、オペレーティングシステムの略であり、Linux 等が一例である。OS 710 は、他のサブプログラムを平行して実行するカーネル 711 およびライブラリ 712 等で構成されるプログラム群の総称であり、詳細な説明は省略する。本構成例においては、OS 710 のカーネル 711 は、電源投入後にナビゲータ 720、Java ミドルウェア 740、HTML ブラウザ 750、およびメーラー 760 をサブプログラムとして起動する。OS 710 が Linux である場合、OS 710 は、これらサブプログラムに対してそれぞれ個別のプロセスを起動し、それぞ

れのプロセス内で個々のサブプログラムを実行する。

OS 710のライブラリ712は、他のサブプログラムに対して、デジタルテレビ100が有するハードウェア構成要素（リソース）を制御するための複数の機能を提供する。つまり、図4に示すリソース制御部47は、ライブラリ712の機能の一部として実現される。本構成例では、複数の機能の一例として、チューナライブラリ（チューナL i b）7121、デスクランプラライブラリ（デスクランプラL i b）7122、およびAV再生ライブラリ（AV再生L i b）7123を取り上げる。これらは、図4に示すリソース制御部47の機能を実現するものである。

チューナL i b 7121は、チューナ101を制御する機能をサブプログラムに提供する。すなわち、チューナL i b 7121は、チューナに与える周波数を含むチューニング情報をサブプログラムから受け取る。そして、チューナL i b 7121は、受け取った情報をチューナ101に与えるとともに、チューニングを行うようにチューナ101に指示する。

デスクランプラL i b 7122は、デスクランプラ102を制御する機能をサブプログラムに提供する。すなわち、デスクランプラL i b 7122は、鍵等を含む復号情報をサブプログラムから受け取る。そして、デスクランプラL i b 7122は、受け取った情報をデスクランプラ102に与えるとともに、デスクランプラ102に復号を行うように指示する。

AV再生L i b 7123は、TSデコーダ103を制御

し、映像・音声再生を行う機能をサブプログラムに提供する。すなわち、AV再生Lib7123は、再生すべき映像のパケットIDおよび音声のパケットIDをサブプログラムから受け取る。そして、AV再生Lib7123は、受け取った映像のパケットIDと出力先「ビデオデコーダ106」との組、および、受け取った音声のパケットIDと出力先「オーディオデコーダ104」との組をTSデコーダ103に与えると共に、TSデコーダ103にフィルタリングを行うように指示する。これによって、音声データを含むパケットがオーディオデコーダ104に引き渡された後、音声データはスピーカ105を通して再生される。また、映像データを含むパケットがビデオデコーダ106に引き渡された後、映像データは、ディプレイ107を通して再生される。

ナビゲータ720は、例えばユーザの指示に従って、フォーマットエンジンの起動や停止をフォーマットエンジンに対して指示する。また、ナビゲータ720は、デジタルテレビのチャンネル切り替えや、フォーマットエンジンでのアプリケーションの実行・停止、データの再生・停止等を指示する。ナビゲータ720は、OS710のカーネル711によって起動されるサブプログラムである。

次に、ナビゲータ720によるチャンネルの切り替えの動作を説明する。なお、入力部111は、図3で示されるフロントパネルであるとする。ユーザがC+ボタン308あるいはC-ボタン309を押下すると、ナビゲータ720は、押下されたボタンの識別子を受け取り、再生してい

るチャンネルを切り替える。ここで、C+ボタン308はチャンネル番号を番号が大きくなる方へ変化させ、C-ボタン309はチャンネル番号を番号が小さくなる方へ変化させる。再生可能なチャンネルの番号が例えば「1」、「2」「4」「6」「8」で、現在チャンネル「6」を再生しているとする。ここで、ユーザがC+ボタン308を押下するとナビゲータ720は、再生するチャンネルを「6」から「8」に変更する。また、ユーザがC-ボタン309を押下するとナビゲータ720は、再生するチャンネルを「6」から「4」に変更する。チャンネル変更は、チャンネルに対応する映像・音声を含むパケットIDをOS710のライブラリ712に含まれるAV再生Lib7123に与えることによって行われる。この動作は、現在市販されているテレビの基本的な機能であるので、詳細な説明は省略する。

次に、図8～図16を用いて、フォーマットエンジン等の起動指示がユーザによって行われる際のナビゲータの動作を説明する。ユーザが入力部111のMENUボタン307を押下すると、ナビゲータ720は、実行可能なフォーマットエンジンの一覧をディスプレイ107に表示する。図8、図9および図10は、ディスプレイ107に表示されるフォーマットエンジンの一覧の例を示す図である。図8～図10では、フォーマットエンジンの一覧として、フォーマットエンジン名とその状態との組が表示される。さらに、フォーマットエンジンを選択するためにカーソル81が表示される。カーソル81は、ユーザが入力部11

1 の上カーソルボタン 301 または下カーソルボタン 302 を押下することに応じて移動する。図 8 に示す状態で下カーソルボタン 302 が押下されると、カーソル 81 は 1 つ下に移動して図 9 に示す状態になる。図 9 の状態で下カーソルボタン 302 が押下されると、カーソル 81 は 1 つ下に移動して図 10 に示す状態になる。図 9 に示す状態で上カーソルボタン 301 が押下されると、カーソル 81 は 1 つ上に移動して図 8 に示す状態になる。なお、図 8 ～図 10 において、表示される各フォーマットエンジン (Java ミドルウェア 740、HTML ブラウザ 750、およびメーラー 760) の状態は、ナビゲータ 720 によって結合部 730 から取得される。

また、図 8 に示す状態でユーザが入力部 111 のOKボタン 305 を押下すると、ナビゲータ 720 は、Java ミドルウェア 740 が実行可能なアプリケーションの一覧を表示する。図 11 および図 12 は、ディスプレイ 107 に表示される、実行可能なアプリケーションの一覧の例を示す図である。図 11 および図 12 では、実行可能なアプリケーションの一覧として、アプリケーション名とその状態との組が表示される。さらに、アプリケーションを選択するためにカーソル 81 が表示される。カーソル 81 は、ユーザが入力部 111 の上カーソルボタン 301 または下カーソルボタン 302 を押下することに応じて移動する。図 11 の状態で下カーソルボタン 302 が押下されると、カーソル 81 は 1 つ下に移動して図 12 の状態になる。図 12 の状態で上カーソルボタン 301 が押下されると、カ

ーソル 8 1 は 1 つ上に移動して図 1 1 の状態になる。なお、図 1 1 および図 1 2 において、表示される各アプリケーションの状態は、ナビゲータ 7 2 0 によって結合部 7 3 0 から取得される。

また、図 1 1 の状態でユーザがOKボタン 3 0 5 を押下すると、ナビゲータ 7 2 0 は、アプリケーション「E P G」を起動するように結合部 7 3 0 に対して指示する。図 1 2 の状態でOKボタン 3 0 5 を押下すると、ナビゲータ 7 2 0 は、アプリケーション「トランプゲーム」を起動するように結合部 7 3 0 に対して指示する。

図 1 1 および図 1 2 では、全てのアプリケーションが停止中であったが、アプリケーションが「一時停止中」あるいは、「実行中」の場合もある。図 1 3 および図 1 4 は、ディスプレイ 1 0 7 に表示される、実行可能なアプリケーションの一覧の他の例を示す図である。図 1 3 および図 1 4 では、「E P G」の動作状態が「実行中」であり、「トランプゲーム」の動作状態が「一時停止中」である。図 1 3 の状態でユーザが入力部 1 1 1 のOKボタン 3 0 5 を押下すると、ナビゲータ 7 2 0 は、結合部 7 3 0 にアプリケーション「E P G」を停止するように指示する。図 1 4 の状態でユーザが入力部 1 1 1 のOKボタン 3 0 5 を押下すると、ナビゲータ 7 2 0 は、結合部 7 3 0 にアプリケーション「トランプゲーム」を停止するように指示する。まとめると、「停止中」であるアプリケーションをカーソル 1 1 0 によって選択している状態で、ユーザが入力部 1 1 1 のOKボタン 3 0 5 を押下すると、ナビゲータ 7 2 0 は、

当該アプリケーションを起動するように結合部 730 に対して指示する。一方、「実行中」あるいは「一時停止中」であるアプリケーションをカーソル 81 によって選択している状態で、ユーザが入力部 111 のOKボタン 305 を押下すると、ナビゲータ 720 は、結合部 730 にそのアプリケーションを停止するように指示する。

また、図 9 の状態で、ユーザが入力部 111 のOKボタン 305 を押下すると、ナビゲータ 720 は、H T M L ブラウザ 750 が表示可能なデータの一覧を表示する。図 15 および図 16 では、実行可能なデータの一覧として、データ名とその状態との組が表示される。さらに、ユーザがデータを選択するためにカーソル 81 が表示される。カーソル 81 は、ユーザが入力部 111 の上カーソルボタン 301 または下カーソルボタン 302 を押下することにより移動する。図 15 の状態で下カーソルボタン 302 が押下されると、カーソル 81 は 1 つ下に移動して図 16 の状態になる。図 16 の状態で上カーソルボタン 301 が押下されると、カーソル 81 は 1 つ上に移動して図 15 の状態になる。なお、表示される各データの状態は、ナビゲータ 720 によって結合部 730 から取得される。また、図 15 の状態でユーザがOKボタン 305 を押下すると、ナビゲータ 720 は、結合部 730 にデータ「天気予報」を停止するように指示する。図 16 の状態でユーザがOKボタン 305 を押下すると、ナビゲータ 720 は、結合部 730 にデータ「レジャー情報」を停止するように指示する。

また、図 10 の状態でユーザが入力部 111 のOKボタ

ン 305 を押下すると、ナビゲータ 720 は、メーラー 760 を起動するように結合部 730 に対して指示する。メーラー 760 は、取り扱う個々のデータに関する動作状態（例えば、停止中、実行中、一時停止中）を持っておらず、メーラー 760 自身の動作状態のみが規定されている。従って、図 10 の状態でユーザが入力部 111 のOKボタン 305 を押下しても、Java ミドルウェア 740 や HTML ブラウザ 750 のように個々のフォーマットエンジンが取り扱うアプリケーションやデータの一覧を表示することはない。ナビゲータ 720 は、メーラー 760 が取り扱うアプリケーションやデータの状態の一覧を結合部 730 から取得する際、メーラー 760 が個々のアプリケーションやデータの動作状態を保持していないことを知る。あるいは、ナビゲータ 720 は、メーラー 760 が個々のアプリケーションやデータの動作状態を保持していないことを予め知っていてもよい。

以上のように、ナビゲータ 720 は、個々のアプリケーションやデータの動作状態を保持していないフォーマットエンジンに対しては、ナビゲータ 720 は、次のように動作する。すなわち、フォーマットエンジンの動作状態が「停止中」であるフォーマットエンジンがカーソル 81 によって選択されている状態でユーザが入力部 111 のOKボタン 305 を押下すると、ナビゲータ 720 は、当該フォーマットエンジンを起動するように結合部 730 に対して指示する。一方、「実行中」あるいは「一時停止中」であるフォーマットエンジンがカーソル 81 によって選択され

ている状態でユーザが入力部 111 のOKボタン 305 を押下すると、ナビゲータ 720 は、当該フォーマットエンジンを停止するように結合部 730 に対して指示する。

なお、フォーマットエンジンあるいはフォーマットエンジンが取り扱うアプリケーションやデータの一覧が表示されている場合（図 8～図 16）において、ユーザが入力部 111 のMENUボタン 307 を押下すると、ナビゲータ 720 は、当該一覧の表示を取りやめる。

図 4 の説明に戻り、結合部 730 は、ナビゲータ 720 と複数のフォーマットエンジンとを結合するためのモジュールである。結合部 730 は、通信部 731 と、状態管理部 732 と、リソース管理部 733 とを備えている。

通信部 731 は、ナビゲータ 720 と各フォーマットエンジンとの間の通信を行う。ここで、通信とはメッセージデータの送受信を意味する。具体的には、ナビゲータ 720 がフォーマットエンジンに対して指示を行う場合、ナビゲータ 720 は当該指示を示すメッセージを通信部 731 を介して当該フォーマットエンジンへ送る。例えば、ナビゲータ 720 が、Java ミドルウェア 740 が実行しているアプリケーションを停止する旨の指示を行う場合、ナビゲータ 720 は、当該停止の指示を意味するメッセージを通信部 731 を通して Java ミドルウェア 740 に送る。

なお、ナビゲータ 720 や各フォーマットエンジンは、異なるプロセスあるいはスレッドで動作しているので、プロセス間通信あるいはスレッド間通信を行う必要がある。

プロセス間通信あるいはスレッド間通信は、OSがLinuxであれば、Socket、PIPE等のOS710が備える機能を利用することによって実現することができる。

ここで、もし、ナビゲータ720と各フォーマットエンジンとの間で個別に通信方法を決定することとすると、ナビゲータ720および各フォーマットエンジンは、複数種類の通信方法を実装しなければならない。従って、全プログラムのコード量が増え、開発工数も大きくなる。そこで、通信部731は、ナビゲータ720および各フォーマットエンジンが使用する共通の通信方法を規定する。ここでは、ナビゲータ720はSocket通信を用いることとするが、FIFO等、他の方法を用いても実現可能である。通信部731は、さらにSocket通信のプロトコル上に、独自のメッセージフォーマットを規定する。

次に、図17～図31を用いて、メッセージフォーマットについて詳細を説明する。図17は、メッセージフォーマットの一例を示す図である。図17において、メッセージは5つのフィールドで構成される。Source IDフィールド171は、メッセージの送信元であるサブプログラムを示す情報を格納する。なお、サブプログラムには、サブプログラムを識別するためのサブプログラムIDが割り当てられる（後述する図18参照。）。Source IDフィールド171は、メッセージの送信元であるサブプログラムのサブプログラムIDを格納する。具体的には、Source IDフィールド171は、メッセージ

を発信したナビゲータ 720 やフォーマットエンジンを示すサブプログラム I D を 1 バイトで格納する。サブプログラム I D の詳細な内容は後述する。

Destination I D フィールド 172 は、メッセージを受け取るサブプログラムを示すサブプログラム I D (S I D) を 1 バイトで格納する。フォーマットエンジン I D の詳細な内容は後述する。Message I D フィールド 173 は、メッセージの内容を表すメッセージ I D を 1 バイトで格納する。メッセージ I D の詳細な内容は後述する。Data Length フィールド 174 は、Data フィールド 175 に格納されるデータの長さを 2 バイトで格納する。Data フィールド 175 は、メッセージ I D に対応する詳細データを格納する。格納するデータのフォーマットはメッセージ I D に対応して規定される。

図 18 は、図 17 に示す Source I D フィールド 171 および Destination I D フィールド 172 に格納されるサブプログラム I D の表の一例を示す図である。図 18 に示す表では、左側の列にサブプログラムが記載され、右側の列に対応するサブプログラム I D が記載されている。なお、左側の列にある「ALL」は、全てのフォーマットエンジンを示す。ここでは、「ALL」の他、ナビゲータ 720 、結合部 730 、 Java ミドルウェア 740 、 HTML ブラウザ 750 、およびメーラー 760 に対してサブプログラム I D が付されている。つまり、これらのサブプログラムがメッセージのやり取りを行う

サブプログラムである。

図19は、Message IDフィールド173に格納されるメッセージIDの表の一例を示す図である。図19においては、左側の列にメッセージ内容が記載され、中央の列にメッセージID(MID)が記載され、右側の列にDataフィールド175に格納するデータのフォーマット-IDが記載されている。以下、各メッセージの詳細について説明する。

図19において、上から2番目の行は、フォーマットエンジンに対して、フォーマットエンジンの状態を通知するように要求するための「フォーマットエンジン状態要求」メッセージが記載されている。例えば、ナビゲータ720が各フォーマットエンジンの動作状態を知りたいときに使用する。このメッセージはデータフィールド175を使用しない。

図19において、上から3番目の行は、「フォーマットエンジン状態要求」の応答を行うための「フォーマットエンジン状態応答」メッセージである。例えば、Javaミドルウェア740がナビゲータ720から「フォーマットエンジン状態要求」を受け取ると、Javaミドルウェア740は、この「フォーマットエンジン状態応答」メッセージを用いてフォーマットエンジンの状態を通知する。このメッセージは、データフィールド175にフォーマットエンジンの状態を格納する。図20は、フォーマットエンジン状態応答メッセージにおけるデータフィールド175のフォーマットの一例を示す図である。データフィールド

175には、フォーマットエンジンの動作状態を示す動作状態IDが1バイトで格納される。

図2.1は、フォーマットエンジンの動作状態と動作状態IDとの対応を示す表の一例を示す図である。図2.1において、左側の列にフォーマットエンジンの動作状態が記載され、中央の列に動作状態の意味が記載され、右側の列に動作状態IDが記載されている。ここでは、「実行中」、「一時停止中」、および「停止中」という3つの動作状態が規定されている。図2.1において規定されているこれらの動作状態は、全フォーマットエンジンに共通する共通状態である。つまり、メッセージに含まれる動作状態は、共通状態の表現で表されている。また、「実行中」という共通状態は、デジタルテレビ100が有している極小リソースを使用していることを意味する。

図1.9の説明に戻り、上から4番目の行は、フォーマットエンジンの状態が変化したことを、フォーマットエンジンがナビゲータ720や他のフォーマットエンジンに通知するための「フォーマットエンジン状態変化」メッセージである。このメッセージはデータフィールド175を使用しない。

また、上から5番目から7番目の行は、ナビゲータ720等が他のフォーマットエンジンに対して状態を「実行中」「停止中」「一時停止中」に変更するように指示を出すための「フォーマットエンジン実行」「フォーマットエンジン停止」「フォーマットエンジン一時停止」メッセージである。これらのメッセージはデータフィールド175を

使用しない。

また、上から 8 番目の行は、フォーマットエンジンに対して、フォーマットエンジンが実行可能なアプリケーションあるいは表示可能なデータの一覧を通知するように要求するための「アプリケーション・データ一覧要求」メッセージである。例えば、Java ミドルウェア 740 が実行可能なアプリケーションの一覧を、ナビゲータ 720 が知りたい場合に使用される。このメッセージはデータフィールド 175 を使用しない。

また、上から 9 番目の行は、「アプリケーション・データ一覧要求」の応答を行うための「アプリケーション・データ一覧応答」メッセージである。例えば、Java ミドルウェア 740 がナビゲータ 720 から「アプリケーション・データ一覧要求」を受け取ると、Java ミドルウェア 740 は、「アプリケーション・データ一覧応答」メッセージを用いて、Java ミドルウェア 740 が実行可能なアプリケーションとその状態との一覧を通知する。このメッセージはデータフィールド 175 にアプリケーション・データの一覧を格納する。

図 22 は、アプリケーション・データ一覧応答メッセージにおけるデータフィールド 175 のフォーマットの一例を示す図である。図 22 においては、アプリケーション数（図 22 では「アプリ数」と記載している）フィールド 2201 に、フォーマットエンジンが実行可能なアプリケーションあるいは表示可能なデータの数が 1 バイトで格納される。

データフィールド 175 には、アプリケーション数フィールド 2201 に統いて、アプリケーション数フィールド 2201 に格納された数のアプリケーション情報フィールド 2202 が設けられる。

アプリケーション情報フィールド 2202 の先頭には、アプリケーション ID フィールド 2203 が設けられる。アプリケーション ID フィールド 2203 には、実行可能なアプリケーションあるいは表示可能なデータを識別するアプリケーション ID が 1 バイトで格納される。このアプリケーション ID は、フォーマットエンジンが割り当てる。

統いて、アプリケーション情報フィールド 2202 には、アプリケーション状態（図 22 では「アプリ状態」と記載している）ID フィールド 2204 が設けられる。アプリケーション状態 ID フィールド 2204 には、アプリケーションあるいはデータの状態を表すアプリケーション状態 ID を 1 バイトで格納する。なお、本構成例では、アプリケーションの状態とアプリケーション状態 ID との対応は、図 21 に示す対応、すなわち、フォーマットエンジンの動作状態と動作状態 ID との対応と同じであるとする。

統いて、アプリケーション情報フィールド 2202 には、アプリケーション名長さ（図 22 では「アプリ名の長さ」と記載している）フィールド 2205 が設けられる。アプリケーション名長さフィールド 2205 には、アプリケーションあるいはデータの名前の長さが格納される。

アプリケーション情報フィールド 2202 の末尾には、

アプリケーション名（図22では「アプリ名」と記載している）フィールド2206が設けられる。アプリケーション名フィールド2206には、アプリケーション名長さフィールド2205に格納されたデータにより指定される長さのアプリケーションの名前が格納される。

図19の説明に戻り、上から10番目の行は、フォーマットエンジンが実行中のアプリケーションあるいは表示中のデータの状態が変化したことを、フォーマットエンジンがナビゲータ720や他のフォーマットエンジンに通知するための「アプリケーション・データ一覧変化」メッセージである。このメッセージはデータフィールド175を使用しない。

また、上から11番目から13番目の行は、ナビゲータ720等が他のフォーマットエンジンが実行可能なアプリケーションあるいは表示可能なデータに対して状態を「実行中」「停止中」「一時停止中」に変更するように指示を出すための「アプリケーション・データ実行」「アプリケーション・データ停止」「アプリケーション・データ一時停止」メッセージである。これらのメッセージはデータフィールド175に状態を変化させる対象のアプリケーションあるいはデータを識別するためのアプリケーションIDを格納する。

図23は、「アプリケーション・データ実行」「アプリケーション・データ停止」「アプリケーション・データ一時停止」のメッセージにおけるデータフィールド175のフォーマットの一例を示す図である。データフィールド1

75には、アプリケーションあるいはデータを識別するためのアプリケーションID（図23では、「アプリID」と記載している）が1バイトで格納される。アプリケーションIDは、「アプリケーション・データ一覧応答」メッセージでフォーマットエンジンが返す値を用いる。

図19の説明に戻り、上から14番目の行は、ナビゲータ720や他のフォーマットエンジン間で、独自データの送受信時に使用する「プライベート」メッセージである。データフィールド175は、ナビゲータ720や他のフォーマットエンジン間で独自に規定したフォーマットに独自のデータを埋め込んで使用される。

なお、本構成例では、フォーマットエンジンの動作状態の規定と、アプリケーション・データの動作状態の規定とが同じであるが、異なっていても実施可能である。また、本構成例で規定しているメッセージおよびメッセージのフォーマットは一例であり、これ以外のメッセージを含む、他のフォーマットを使用する、あるいは本構成例で示したメッセージの一部・全部を使用しない等してもよい。重要なポイントは、結合部730あるいは同等の構成要素が、ナビゲータ720や複数のフォーマットエンジン間で使用される、フォーマットエンジンの共通状態およびアプリケーション・データの共通状態を規定し、共通の通信手段を提供することにある。

次に、ナビゲータ720の動作に対応するメッセージの具体例を説明する。まず、ナビゲータ720が図8に示すようなフォーマットエンジンの一覧を表示する際には、ナ

ビゲータ 720 は、図 24～図 26 に示す 3 つの「フォーマットエンジン状態要求」メッセージを結合部 730 に渡す。図 24～図 26 は、「フォーマットエンジン状態要求」メッセージの具体例を示す図である。図 24 に示すメッセージ 2400 は、ナビゲータ 720 から Java ミドルウェア 740 への「フォーマットエンジン状態要求」である。Source ID フィールド 2401 にはナビゲータ 720 を表すサブプログラム ID として「1」が格納される（図 18 参照）。Destination ID フィールド 2402 には、Java ミドルウェア 740 を表すサブプログラム ID として「2」が格納される（図 18 参照）。Message ID フィールド 2403 には、「フォーマットエンジン状態要求」を表す「1」が格納される（図 19 参照）。Data Length フィールド 1504 には、データの長さとして「0」が格納される。図 25 に示すメッセージ 2500 は、ナビゲータ 720 から HTML ブラウザ 750 への「フォーマットエンジン状態要求」である。メッセージ 2400 とメッセージ 2500 の違いは、Destination ID フィールド 1512 に、HTML ブラウザ 750 を表す ID 「3」を格納している（図 18 参照）部分だけであり、他は同じである。図 26 に示すメッセージ 2600 は、ナビゲータ 720 からメーラー 760 への「フォーマットエンジン状態要求」である。メッセージ 2400 とメッセージ 2600 の違いは、Destination ID 1522 に、メーラー 760 を表す ID 「4」を格納している（図 18 参照）

部分だけであり、他は同じである。

「フォーマットエンジン状態要求」メッセージが送信された後、ナビゲータ720は、結合部730から図27～図29に示す3つの「フォーマットエンジン取得応答」メッセージを受け取る。図27～図29は、「フォーマットエンジン取得応答」メッセージの具体例を示す図である。メッセージ2700は、Javaミドルウェア740からナビゲータ720への「フォーマットエンジン取得応答」である。Source IDフィールド2701には、Javaミドルウェア740を表すサブプログラムID「2」が格納される（図18参照）。Destination IDフィールド2702には、ナビゲータ720を表すサブプログラムIDとして「1」が格納される（図18参照）。Message IDフィールド2703には、「フォーマットエンジン取得応答」を表すメッセージIDとして「2」が格納される（図19参照）。Data Lengthフィールド2704にはデータの長さ「1」が格納される。データフィールド2705には、「停止中」を表す動作状態IDとして「3」が格納される（図21参照）。メッセージ2800は、HTMLブラウザ750からナビゲータ720への「フォーマットエンジン取得応答」である。メッセージ2700とメッセージ2800の違いは、Source IDフィールド2801に、HTMLブラウザ750を表すサブプログラムIDとして「3」が格納されている部分だけであり、他は同じである。メッセージ2720は、メーラー760からナビゲータ720へ

の「フォーマットエンジン取得応答」である。メッセージ 2700 とメッセージ 2900 の違いは、Source ID フィールド 2901 に、図 18 を参照してマーク 760 を表すサブプログラム ID として「4」が格納されている部分だけであり、他は同じである。このように 3 つの「フォーマットエンジン状態要求」メッセージを結合部 730 に送り、3 つの「フォーマットエンジン取得応答」メッセージを結合部 730 から受け取ることによって、ナビゲータ 720 は、図 8 に示すフォーマットエンジン一覧を表示することができる。

次に、ナビゲータ 720 が図 15 に示すような HTML ブラウザ 750 が表示可能なデータの一覧を表示する場合について説明する。この場合、ナビゲータ 720 は、図 30 に示す「アプリケーション・データ一覧要求」メッセージを結合部 730 に渡す。図 30 は、「アプリケーション・データ一覧要求」メッセージの具体例を示す図である。図 30 に示すメッセージ 3000 は、ナビゲータ 720 から HTML ブラウザ 750 への「アプリケーション・データ一覧要求」である。Source ID フィールド 1701 には、ナビゲータ 720 を表すサブプログラム ID として「1」が格納される（図 18 参照）。Destination ID フィールド 1702 には、HTML ブラウザ 750 を表すサブプログラム ID として「3」が格納される（図 18 参照）。Message ID フィールド 1703 には、「アプリケーション・データ一覧要求」を表すメッセージ ID として「11」が格納される（図 19 参

照）。Data Length フィールド 1704 にはデータの長さを示す「0」が格納される。

「アプリケーション・データ一覧要求」メッセージを送信した後、ナビゲータ 720 は、結合部 730 から図 31 に示す「アプリケーション・データ取得応答」メッセージを受け取る。図 31 は、「アプリケーション・データ取得応答」メッセージの具体例を示す図である。図 31 に示すメッセージ 3100 は、HTML ブラウザ 750 からナビゲータ 720 への「アプリケーション・データ取得応答」である。Source ID フィールド 3101 には、HTML ブラウザ 750 を表すサブプログラム ID として「3」が格納される（図 18 参照）。Destination ID フィールド 3102 には、ナビゲータ 720 を表すサブプログラム ID として「1」が格納される（図 18 参照）。Message ID 3 フィールド 103 には、「アプリケーション・データ取得応答」を表すメッセージ ID として「12」が格納される（図 19 参照）。Data Length フィールド 1604 には、データの長さを示す「27」が格納される。アプリケーション数フィールド 3105 には、HTML ブラウザ 750 が表示可能なデータの数を示す「2」が格納される。

また、図 31において、アプリケーション数フィールド 3105 には、1 つ目のデータに対するアプリケーション情報（図 31 には、「アプリ情報」と記載している）フィールド 3111 と、2 つ目のデータに対するアプリケーション情報（図 31 には、「アプリ情報」と記載している）

フィールド 3 1 1 2 とが含まれている。アプリケーション I D フィールド 3 1 2 1 には、1 つ目のデータを表すアプリケーション I D として「1」が格納されている。アプリケーション名フィールド 3 1 2 4 には、データの名前を示す「天気予報」が格納されている。ここで「天気予報」の各文字は 2 バイトコードで表現されており、結果、アプリケーション名長さは、4 文字 × 2 バイト = 8 バイトとなっている。アプリケーション I D フィールド 3 1 2 5 には 2 つ目のデータを表すアプリケーション I D として「2」が格納されている。状態 I D フィールド 3 1 2 7 には、「実行中」を表すアプリケーション状態 I D として「3」が格納されている。アプリケーション名長さフィールド 3 1 2 8 には、アプリケーション名の長さを示す「1 2」が格納される。アプリケーション名フィールド 3 1 2 4 には、データの名前を示す「レジャー情報」が格納されている。ここで「レジャー情報」の各文字は 2 バイトコードで表現されており、結果、アプリケーション名長さは、6 文字 × 2 バイト = 1 2 バイトとなっている。

以上で、メッセージフォーマットについての説明を終了する。

再び図 4 の説明に戻り、通信部 7 3 1 は、ナビゲータ 7 2 0 またはフォーマットエンジンからメッセージを受け取ると、メッセージに含まれている Destination I D フィールドを調べ、対応するフォーマットエンジンあるいはナビゲータ 7 2 0 に当該メッセージを配達する。例えば、Destination I D フィールドに「0

」が格納されている場合、通信部 731 は、全てのフォーマットエンジンおよびナビゲータ 720 にメッセージを送信する（図 18 参照）。従って、ナビゲータ 720 は、全てのフォーマットエンジンを終了させたい場合、Destination ID フィールドが「0」であり、メッセージ ID が「5」であるメッセージを通信部 31 へ送ればよい。

状態管理部 732 は、フォーマットエンジンの動作状態、およびフォーマットエンジンが実行可能なアプリケーションおよび表示可能なデータの動作状態を保持する。具体的には、状態管理部 732 は、それらの情報を 2 次記憶部 108 あるいは 1 次記憶部 109 に記憶する。

図 32～図 34 は、状態管理部 732 が保持している情報の一例を示す図である。図 32 は、フォーマットエンジンに関する情報を表形式で表した例である。図 32 において、最も左側の列は、各フォーマットエンジンを識別するためのフォーマットエンジン ID である。ここでは、各フォーマットエンジンのフォーマットエンジン ID は、図 18 で規定されたサブプログラム ID と同じであるとする。左から 2 番目の列は、各フォーマットエンジンの名前である。左から 3 番目の列は、各フォーマットエンジンの動作状態を表す動作状態 ID である。図 32 に示す表において、動作状態 ID は、図 21 において規定される値によって表される。最も右側の列は、各フォーマットエンジンが実行可能なアプリケーションあるいは表示可能なデータに関する情報の格納位置を示すポインターである。実行可能な

アプリケーションあるいは表示可能なデータを持たないフォーマットエンジンに関しては、当該列は「NULL」と記述される。

また、図32に示す表において上から2番目の行には、Javaミドルウェア740の情報が記述されている。すなわち、フォーマットエンジンIDが「2」、名前が「Javaミドルウェア」、動作状態IDが、「停止中」を意味する「3」、実行可能なアプリケーションに関する情報に対するポインターが16進数表記で「0x5678」と記述されている。上から3番目の行には、HTMLブラウザ750の情報が記述されている。すなわち、フォーマットエンジンIDが「3」、名前が「HTMLブラウザ」、動作状態IDが、「停止中」を意味する「3」、表示可能なデータに関する情報に対するポインターが16進数表記で「0x7162」と記述されている。上から4番目の行には、メーラー760の情報が記述されている。すなわち、フォーマットエンジンIDが「4」、名前が「メーラー」、動作状態IDが、「停止中」を意味する「3」と記述されている。また、アプリケーション・データに関する情報はないので、ポインターが「NULL」と記述されている。

図33は、Javaミドルウェア740が実行可能なアプリケーションに関する情報を表形式で表した例を示す図である。この情報は、1次記憶部109中の格納位置0x5678から格納されている。図33において左側の列には、各アプリケーションに割り当てられたアプリケーショ

ン I D が記述される。中央の列には、アプリケーションの名前が記述される。右側の列には、アプリケーション状態 I D が記述される。上から 2 番目の行には、アプリケーション I D が「1」、名前が「E P G」、アプリケーション状態 I D が、「停止中」を表す「3」と記述されている。上から 3 番目の行には、アプリケーション I D が「2」、名前が「トランプゲーム」、アプリケーション状態 I D が、「停止中」を表す「3」と記述されている。

図 3 4 は、H T M L ブラウザ 7 5 0 が実行可能なデータに関する情報を表形式で表した例を示す図である。この情報は、1 次記憶部 1 0 9 中の格納位置 0 x 7 1 6 2 から格納されている。図 3 4 において左側の列には、各データに割り当てられたデータ I D が記述されている。中央の列には、データの名前が記述されている。右側の列には、データの状態を表すアプリケーション状態 I D が記述されている。上から 2 番目の行には、アプリケーション I D が「1」、名前が「天気予報」、アプリケーション状態 I D が、「実行中」を表す「1」と記述されている。上から 3 番目の行には、アプリケーション I D が「2」、名前が「レジャー情報」、アプリケーション状態 I D が、「実行中」を表す「1」と記述されている。

再び図 4 の説明に戻る。上述したように、通信部 7 3 1 は、メッセージに含まれる D e s t i n a t i o n I D フィールドに格納されたデータに従ってメッセージを配達する。ここで、「フォーマットエンジン状態要求」メッセージおよび「アプリケーション・データ一覧要求」メッセージ

ージに関しては、通信部 731 は、Destination ID フィールドを無視して、状態管理部 732 に配達してもよい。状態管理部 732 は、各フォーマットエンジンの状態を保持しているので、各フォーマットエンジンに代わって「フォーマットエンジン状態要求」メッセージ、「フォーマットエンジン状態応答」メッセージ、あるいは「アプリケーション・データ一覧応答」メッセージを生成することができる。状態管理部 732 がこれらのメッセージを生成することによって、プロセス間通信やスレッド間通信を省略することができるので、短時間での応答を実現することができる。

状態管理部 732 は、常に最新のフォーマットエンジンの動作状態を保持する。従って、状態管理部 732 は、必要に応じて各フォーマットエンジンから情報を取得する。すなわち、まず、デジタルテレビ 100 に電源が投入されると、状態管理部 732 は、各フォーマットエンジンに対して「フォーマットエンジン状態要求」メッセージを送る。そして、状態管理部 732 は、各フォーマットエンジンから「フォーマットエンジン状態応答」メッセージを受け取り、受け取ったメッセージに基づいて各フォーマットエンジンの動作状態を 1 次記憶部 109 に記憶する。また、状態管理部 732 は、各フォーマットエンジンに対して「アプリケーション・データ一覧要求」メッセージを送る。そして、状態管理部 732 は、各フォーマットエンジンから「アプリケーション・データ一覧応答」メッセージを受け取り、受け取ったメッセージに基づいて各アプリケーシ

ヨンおよび各データのアプリケーション状態を1次記憶部109に記憶する。

さらに、状態管理部732は、フォーマットエンジンから「フォーマットエンジン状態変化」メッセージを受け取ると、当該フォーマットエンジンに対して「フォーマットエンジン状態要求」メッセージを送る。そして、状態管理部732は、メッセージを送ったフォーマットエンジンから「フォーマットエンジン状態応答」メッセージを受け取り、受け取ったメッセージに基づいて各フォーマットエンジンの動作状態を1次記憶部109に記憶する。

次に、図35～図38を用いて、メーラー760の動作状態が変化する場合におけるメッセージのやり取りを説明する。図35は、メーラー760が送信する「フォーマットエンジン状態変化」メッセージの一例を示す図である。図35に示すメッセージ3500は、メーラー760から全フォーマットエンジンへの「フォーマットエンジン状態変化」メッセージの一例である。メッセージ3500においては、Source IDフィールド3501には、メーラー760を表すサブプログラムIDとして「4」が格納される（図18参照）。Destination IDフィールド3502には、全部を表すサブプログラムIDとして「0」が格納される（図18参照）。Message IDフィールド3503には、動作状態IDとして「フォーマットエンジン状態変化」を表す「3」が格納される（図19参照）。Data Lengthフィールド3504には、データの長さを表す「0」が格納される。

状態管理部 732 は、図 35 に示すメッセージ 3500 を受け取ると、図 36 に示されるメッセージ 3600 をメーラー 760 に送る。図 36 は、状態管理部 732 が送信するメッセージの一例を示す図である。図 36 に示すメッセージ 3600 においては、Source ID フィールド 3601 には、結合部 730 を表すサブプログラム ID として「5」が格納される（図 18 参照）。Destination ID フィールド 3602 には、メーラー 760 を表すサブプログラム ID として「4」が格納される（図 18 参照）。Message ID フィールド 3603 には、メッセージ ID として「フォーマットエンジン状態取得」を表す「1」が格納される（図 19 参照）。Data Length フィールド 3604 には、データの長さを表す「0」が格納される。

メーラー 760 は、図 36 に示すメッセージ 3600 を受け取ると、図 37 で示されるメッセージ 3700 を結合部 730 の状態管理部 732 に送る。図 37 は、メーラー 760 が送信するメッセージの一例を示す図である。図 37 に示すメッセージ 3700 においては、Source ID フィールド 3701 には、メーラー 760 を表すサブプログラム ID として「4」が格納される（図 18 参照）。Destination ID フィールド 3702 には、結合部 730 を表すサブプログラム ID として「5」が格納される（図 18 参照）。Message ID フィールド 3703 には、メッセージ ID として、「フォーマットエンジン取得応答」を表す「2」が格納される（図 19

参照）。Data Length フィールド 3704 には、データの長さを表す「1」が格納される。データフィールド 3705 には、動作状態 ID として「実行中」を表す「1」が格納される（図 21 参照）。

図 37 に示すメッセージ 3700 が結合部 730 の状態管理部 732 へ送信された結果、状態管理部 732 は、メーラー 760 の状態が「実行中」に変化したことを知ることができる。このとき、メッセージ 3700 の受信前の状態が図 32 に示す状態であったとすると、状態管理部 732 は、図 38 に示すように、1 次記憶部 109 に記憶している情報を更新する。図 38 は、図 32 の変化後の状態を示す図である。図 38 では、メーラー 760 の動作状態 ID が「1」になっている。

また、状態管理部 732 は、フォーマットエンジンから「アプリケーション・データ一覧変化」メッセージを受け取ると、「フォーマットエンジン状態変化」メッセージの場合と同様、最新の状態を取り直す。すなわち、状態管理部 732 は、メッセージを送ってきたフォーマットエンジンに対して、「アプリケーション・データ一覧要求」メッセージを送る。そして、状態管理部 732 は、当該フォーマットエンジンから「アプリケーション・データ一覧応答」メッセージを受け取り、受け取ったメッセージに基づいて各アプリケーションおよび各データのアプリケーション状態を 1 次記憶部 109 に記憶する。

以上のようにして、状態管理部 732 は、各フォーマットエンジン、フォーマットエンジンが実行するアプリケ

ション、およびフォーマットエンジンが表示するデータの最新の情報を保持することができる。

次に、図39～図44を用いて、リソース管理部733について説明する。なお、リソース管理部733は、上述した第2の動作例の動作を行う際に必要となる構成であって、第1の動作例の動作を行う場合には不要である。

リソース管理部733は、複数のフォーマットエンジン間で発生したリソースの競合を解決するための情報を提供する。図39は、リソース管理部733の内部構成を示すブロック図である。図39において、リソース管理部733は、プロセス記憶部3901と、優先度記憶部3902と、最新起動記憶部3903と、リソースID記憶部3904と、フォーマットエンジン特定部3905と、優先度情報提供部3906と、リソース剥奪通知部3907とを備えている。

プロセス記憶部3901は、2次記憶部108あるいは1次記憶部109によって実現され、各フォーマットエンジンを実行しているプロセスあるいはスレッドの情報を保持する。ここで、各フォーマットエンジンはプロセス上で実行されているとする。

図40は、プロセス記憶部3901において記憶されている情報の一例を示す図である。図40においては、プロセス記憶部3901は、各フォーマットエンジンを実行しているプロセスIDを表形式で記憶している。図40において、左側の列には、フォーマットエンジンID(FID)が記述されている。右側の列には、各フォーマットエン

ジンを実行しているプロセスを識別するためのプロセス I D が記述されている。プロセス I D は、OS 710 のカーネル 711 によって生成されるプロセスの I D である。リソース管理部 733 は、カーネル 711 からプロセス I D を受け取ってプロセス記憶部 3901 に記憶する。一方、上から 2 番目の行には、Java ミドルウェア 740 のフォーマットエンジン I D である「2」(図 18 参照) と、Java ミドルウェア 740 を実行しているプロセスのプロセス I D である「100」とが記述されている。上から 3 番目の行には、HTML ブラウザ 750 のフォーマットエンジン I D である「3」(図 18 参照) と、HTML ブラウザ 750 を実行しているプロセスのプロセス I D である「110」とが記述されている。上から 4 番目の行には、メーラー 760 のフォーマットエンジン I D である「4」(図 18 参照) と、メーラー 760 を実行しているプロセスのプロセス I D である「120」とが記述されている。

図 39 の説明に戻り、優先度記憶部 3902 は、2 次記憶部 108、1 次記憶部 109 あるいは ROM 110 によって実現される。優先度記憶部 3902 は、各フォーマットエンジンの優先度を記憶する。

図 41 は、優先度記憶部 3902 に記憶されている情報の一例を示す図である。図 41 において、優先度記憶部 3902 は、各フォーマットエンジンの優先度を表形式で記憶している。図 41 において、左側の列には、フォーマットエンジン I D (FID) が記述されている。右側の列に

は、各フォーマットエンジンの優先度が記述されている。ここで、優先度は値が大きいほど優先度が高いとする。また、上から2番目の行には、JavaミドルウェアのフォーマットエンジンIDである「2」と、それに対応する優先度である「2」とが記述されている。上から3番目の行には、HTMLブラウザのフォーマットエンジンIDである「3」と、それに対応する優先度「1」とが記述されている。上から4番目の行には、メールのフォーマットエンジンIDである「4」と、それに対応する優先度「4」とが記述されている。

図3.9の説明に戻り、最新起動記憶部3903は、2次記憶部108あるいは1次記憶部109によって実現される。最新起動記憶部3903は、ナビゲータ720から送られてきた「フォーマットエンジン実行」メッセージ、または「アプリケーション・データ実行」メッセージに含まれているDestination IDフィールドに記述されたフォーマットエンジンIDを記憶する。つまり、最新起動記憶部3903は、最新起動情報を記憶する。最新起動情報とは、ユーザが最後に実行を要求したフォーマットエンジンを示す情報である。

図4.2は、最新起動記憶部3903が記憶している情報の一例を示す図である。図4.2では、最新起動記憶部3903は、メール760のフォーマットエンジンIDである「4」を格納している。図4.2に示す状態において、ナビゲータ720が、結合部730を通してHTMLブラウザ750に「フォーマットエンジン実行」メッセージ、あ

るいはアプリケーション・データ実行」メッセージを送った場合を考える。この場合、リソース管理部 733 は、最新起動記憶部 3903 に記憶される情報を、H T M L ブラウザ 750 を表すフォーマットエンジン I D である「3」に更新する。

図 39 の説明に戻り、リソース I D 記憶部 3904 は、2 次記憶部 108 あるいは R O M 110 によって実現される。リソース I D 記憶部 3904 は、リソース管理部 733 が管理対象とするリソースを識別するためのリソース I D (R I D) を記憶する。

図 43 は、リソース I D 記憶部 3904 が記憶している情報の一例を示す図である。図 43においては、リソース I D 記憶部 3904 は、リソース I D を表形式で記憶する。図 43において、左側の列にはリソース I D が記述されている。右側の列には、対応するリソースの名前が記述されている。図 43 では、管理対象のリソースとして、チューナ 101、T S デコーダ 103、オーディオデコーダ 104、ビデオデコーダ 106、ネットワークインターフェース 112 が規定されている。これらのリソースに対してリソース I D が割り当てられている。図 44 は、リソース I D の定義をプログラミング言語である C 言語で記述した例を示す図である。

図 39 の説明に戻り、フォーマットエンジン特定部 3905 は、リソースを要求しているフォーマットエンジンを特定する。各フォーマットエンジンは、O S 710 のライブラリ 712 が提供する機能を利用する。例えば、チュー

ナ 1 0 1 を制御するチューナ L i b 7 1 2 1 を呼び出し、特定の周波数帯にチューニングを行う。これを実現するために、フォーマットエンジンは、チューナ L i b 7 1 2 1 が用意する関数を呼び出す。フォーマットエンジンから要求があると、すなわち、フォーマットエンジンが関数の呼び出しを行うと、チューナ L i b 7 1 2 1 は、どのフォーマットエンジンが当該関数を呼び出しているのかをフォーマットエンジン特定部 3 9 0 5 に問い合わせせる。この問い合わせに応じて、フォーマットエンジン特定部 3 9 0 5 は、関数を呼び出しているプロセスを特定する。そして、フォーマットエンジン特定部 3 9 0 5 は、特定したプロセスのプロセス I D、およびプロセス記憶部 3 9 0 1 が記憶しているフォーマットエンジン I D とプロセス I D との組を参照することによって、関数を呼び出したフォーマットエンジンを特定する。

優先度情報提供部 3 9 0 6 は、優先度記憶部 3 9 0 2 が記憶しているフォーマットエンジンの優先度と、最新起動記憶部 3 9 0 3 が記憶している、ユーザが最後に起動したフォーマットエンジンの情報をライブラリ 7 1 2 に提供する。ライブラリ 7 1 2 は、フォーマットエンジン特定部 3 9 0 5 および優先度情報提供部 3 9 0 6 を利用することによって、どのフォーマットエンジンにリソースを提供するかを決定することができる。以下、具体的に説明する。

例えば、チューナ L i b 7 1 2 1 は、Java ミドルウェア 7 4 0 から呼び出され、チューニングを行っているとする。なお、チューナ L i b 7 1 2 1 は、フォーマットエ

ンジン特定部 3905 を利用することによって、Java ミドルウェア 740 が現在チューナ 101 を利用していることを知っている。この状態において、メーラー 760 がチューナ Lib 7121 を呼び出し、他の周波数帯にチューニングしようとする場合を考える。チューナ Lib 7121 は、フォーマットエンジン特定部 3905 を利用することによって、メーラー 760 がチューナ 101 の機能を利用していることを知る。次に、チューナ Lib 7121 は、優先度情報提供部 3906 から最後に起動したフォーマットエンジンの情報を取得する。そして、チューナ Lib 7121 は、最後に起動したフォーマットエンジンがメーラー 760 であれば、メーラー 760 にチューナ 101 の利用を許可する。一方、最後に起動したフォーマットエンジンが Java ミドルウェア 740 であれば、メーラー 760 に対してチューナ 101 の利用を拒否する。最後に起動したフォーマットエンジンがメーラー 760 でも Java ミドルウェア 740 でもない場合は、優先度情報提供部 3906 からメーラー 760 および Java ミドルウェア 740 の優先度を取得する。メーラー 760 の優先度が Java ミドルウェア 740 の優先度よりも高い場合、チューナ Lib 7121 は、メーラー 760 にチューナ 101 の使用を許す。一方、メーラー 760 の優先度が Java ミドルウェア 740 の優先度よりも低い場合、メーラー 760 に対して、チューナ 101 の使用を拒否する。以上のようにして、ライブラリ 712 は、どのフォーマットエンジンにリソースを提供するかを決定することができる。

なお、ライブラリ 712 が、最後に起動したフォーマットエンジンの情報や優先度情報をどのように用いるかは自由であり、ここに紹介したのはその一例である。どのフォーマットエンジンに対してリソースの利用を許可するかを決定するためのルールは、どのようなものであってもよい。例えば、ライブラリ 712 は、優先度情報のみを用いてリソースの利用を許可するフォーマットエンジンを決定してもよい。また、ライブラリ 712 は、先に使用しているフォーマットエンジンに対して利用を許可してもよい。逆に、ライブラリ 712 は、後から呼び出したフォーマットエンジンに対して利用を許可してもよい。

図 39において、リソース剥奪通知部 3907 は、リソースの利用が許可されなかつたフォーマットエンジンに対して、リソースが剥奪されたことを通知する。例えば、Java ミドルウェア 740 がチューナ L i b 7121 を呼び出し、チューナ 101 を使用しているとき、メーラー 760 がチューナ L i b 7121 を呼び出し、チューナ 101 の使用を試みた場合を考える。ここで、もし、チューナ L i b 7121 がメーラー 760 にチューナ 101 の使用を許すと、Java ミドルウェア 740 が希望している周波数帯とは異なる周波数帯にチューニングされる。従って、Java ミドルウェア 740 上で実行されているアプリケーションは正しい動作を続けることができなくなるおそれがある。これを防ぐために、チューナ L i b 7121 は、チューナリソースが剥奪されたことを Java ミドルウェア 740 に通知するように、リソース剥奪通知部 390

7に依頼する。この依頼の際には、リソースが剥奪されたことを通知すべきフォーマットエンジンのフォーマットエンジンIDとリソースIDとがリソース剥奪通知部3907に渡される。なお、ライブラリ712は、フォーマットエンジンIDをフォーマットエンジン特定部3905から取得することによって知ることができる。また、ライブラリ712は、リソースID記憶部3904を参照することによってライブラリ712を予め知っている。リソース剥奪通知部3907は、フォーマットエンジンIDおよびリソースIDを受け取ると、フォーマットエンジンIDにより示されるフォーマットエンジンにリソースが剥奪されたことを通知する。なお、通知を受けたフォーマットエンジンの動作は、各フォーマットエンジンに依存する。動作続行が困難なフォーマットエンジンであれば、実行中から停止中あるいは一時停止中に動作状態を変化させようにもよい。以上でリソース管理部733の説明を終了する。

次に、図45～図71を用いて、各フォーマットエンジンの詳細を説明する。

まず、図45～図51を用いて、Javaミドルウェア740について説明する。Javaミドルウェア740は、Javaアプリケーションを実行するフォーマットエンジンである。本構成例においては具体的な例として、DVB-MHP1.0規格（正式には、ETSI TS 101 812 DVB-MHP仕様V1.0.2）を取り上げるが、例えばアプレットビューワ等、他の仕様に従って

動作するものでも実施可能である。本構成例では、Javaミドルウェア740の詳細な動作にはDVB-MHP1.0規格の内容が含まれる。従って、ここでは、概略のみを記述する。

図45は、Javaミドルウェア740の構成を示すブロック図である。Javaミドルウェア740は、Javaバーチャルマシン(VM)4501と、クラスライブラリ4502と、アプリケーションマネージャ4503とを備えている。

Java VM 4501は、Java(TM)言語で記述されたアプリケーションを逐次解析し実行するJavaバーチャルマシンである。Java言語で記述されたプログラムはバイトコードと呼ばれる、ハードウェアに依存しない中間コードにコンパイルされる。Javaバーチャルマシンは、このバイトコードを実行するインターブリタである。また、一部のJavaバーチャルマシンは、バイトコードをCPU113が理解可能な実行形式に翻訳してから、CPU113に引き渡し、実行することも行う。クラスライブラリ4502およびアプリケーションマネージャ4503の一部または全部もJava言語で記述されており、Java VM 4501によって実行される。本構成例では、OS710のカーネル711は、最初に実行するJavaアプリケーションとしてアプリケーションマネージャ4503を指定する。Java言語の詳細は、書籍「Java Language Specification (ISBN 0-201-63451-1)」等の多くの書

籍で解説されている。従って、ここでは、その詳細を省略する。また、Java VM自体の詳細な動作などは、「Java Virtual Machine Specification (ISBN 0-201-63451-X)」等の多くの書籍で解説されている。ここでは、その詳細を省略する。

クラスライブラリ4502は、ROM110に格納されている複数のJavaクラスライブラリの集合である。その多くは、OS710のライブラリ712が提供する機能に対応するJavaアプリケーションが呼び出し可能なJava APIを提供する。この結果、Javaアプリケーションはデジタルテレビ100が有する機能を利用することができる。例えば、DVB-MHP1.0規格では、チューナ101を使用するためのAPIとしてorg.davic.mpeg.tuningパッケージを規定し、Javaアプリケーションがチューナ101を制御できるようにしている。

アプリケーションマネージャ4503は、放送波中に多重化されているJavaアプリケーションをダウンロードし、実行する。MHP1.0規格によれば、AITと呼ばれるJavaアプリケーションを定義する表が放送波中に多重化されて伝送される。アプリケーションマネージャ4503は、このAITを最初にダウンロードする。AITにはアプリケーションをダウンロードするための情報やアプリケーションの名前、アプリケーションの制御情報などが含まれる。

図46は、AITの主要部を表した模式図である。図46において最も左側の列には、JavaアプリケーションIDが記述されている。左から2番目の列には、Javaアプリケーションの制御情報が記述されている。制御情報には「auto start」、「present」、「kill」等がある。「auto start」は、Javaミドルウェア740がそのJavaアプリケーションを即時にかつ自動的に実行することを意味する。「present」は、自動実行しないことを意味する。「kill」は、Javaアプリケーションを停止することを意味する。左から3番目の列は、Javaアプリケーションのアプリケーション名が記述されている。最も右側の列には、Javaアプリケーションの優先度が記述されている。Javaアプリケーション間でリソースの競合が発生した場合、優先度の高いJavaアプリケーションにリソースが優先的に割り当てられる。ここで、優先度の値が大きいほど優先度が高いこととする。

図46において上から2番目および3番目の行には、Javaアプリケーションの情報の組が記述される。上から2番目の行において定義されるJavaアプリケーションについては、JavaアプリケーションIDが「1」であり、制御情報が「auto start」であり、アプリケーション名が「EPG」であり、優先度が「64」である。上から3番目の行において定義されるJavaアプリケーションについては、JavaアプリケーションIDが「2」であり、制御情報が「present」であり、アプ

リケーション名が「トランプゲーム」であり、優先度「32」である。

アプリケーションマネージャ4503は、A I Tを受け取ると、アプリケーション名「E P G」のJavaアプリケーションをダウンロードし実行する。ここで、D V B - M H P 1. 0 規格が規定するJavaアプリケーションは、「L o a d e d」、「P a u s e d」、「A c t i v e」、および「D e s t r o y e d」という4つの動作状態を有している。

図47は、Javaアプリケーションの4つの動作状態および動作状態間の遷移を表す状態遷移図である。アプリケーションマネージャ4503がJavaアプリケーションをダウンロードし、当該Javaアプリケーションが1次記憶部109に格納されると、当該Javaアプリケーションの動作状態は「L o a d e d」になる。この動作状態から他の動作状態への遷移は、ダウンロードしたJavaアプリケーションが実装している「i n i t X l e t」、「s t a r t X l e t」、「p a u s e X l e t」、および「d e s t r o y X l e t」というメソッドをアプリケーションマネージャ4503が呼び出すことにより実現する。

「L o a d e d」の動作状態でアプリケーションマネージャ4503が「i n i t X l e t」メソッドを呼び出すと、Javaアプリケーションの動作状態は、「P a u s e d」に遷移する。「P a u s e d」の動作状態でアプリケーションマネージャ4503が「s t a r t X l e t」

メソッドを呼び出すと、Javaアプリケーションの動作状態は「Active」に遷移する。「Active」の動作状態でアプリケーションマネージャ4503が「pauseXlet」メソッドを呼び出すと、Javaアプリケーションの動作状態は、「Paused」に遷移する。どの動作状態の場合においても、アプリケーションマネージャ4503が「destroyXlet」メソッドを呼び出すと、Javaアプリケーションの動作状態は「Destroyed」に遷移する。

アプリケーションマネージャ4503は、Javaアプリケーションの状態遷移を、(1)放送波中のAITの制御情報、(2)Javaアプリケーションの指示、および(3)ナビゲータ720が結合部730を通して送る指示、の3通りの指示に従って行う。

AITの制御情報が「autostart」の場合、アプリケーションマネージャ4503は、Javaアプリケーションをダウンロードした後、「initXlet」メソッド、「startXlet」メソッドを呼び出し、「Active」状態に遷移させる。AITは、時間と共に変化する。例えば、AITが図46から図48に変化したとする。図48では、「EPG」という名前のアプリケーションの制御情報が「autostart」から「kill」に変更されている。このとき、アプリケーションマネージャ4503は、Javaアプリケーションの「destroyXlet」メソッドを呼び出し、「Destroyed」状態に遷移させる。

一方、Javaアプリケーションは、自分自身の状態や他のJavaアプリケーションの状態を遷移させることができる。DVB-MHP1.0規格では、Javaアプリケーションが自身の状態を遷移させたいときは、Javaアプリケーションは、「`notify Destroyed`」、「`notify Paused`」、および「`resumeRequest`」メソッドのいずれかを呼び出す。「`Destroyed`」状態に遷移するには、Javaアプリケーションは、「`notify Destroyed`」を呼び出す。「`Active`」状態に遷移させるには、Javaアプリケーションは、「`resumeRequest`」メソッドを呼び出す。「`Paused`」状態に遷移するには、Javaアプリケーションは、「`notify Paused`」を呼び出す。他のJavaアプリケーションの状態を遷移させる場合、Javaアプリケーションは、DVB-MHP1.0規格が規定している`org.dvb.application`パッケージが規定するメソッドを利用する。すなわち、まず、他のJavaアプリケーションをアプリケーションIDによって指定した後、Javaアプリケーションは、`org.dvb.application`パッケージが規定するメソッドを呼び出し、指定したJavaアプリケーションの状態を遷移させる。ここでは詳細は省略する。

最後にナビゲータ720からの指示によってJavaアプリケーションの状態を遷移する方法を説明する。ナビゲータ720は、共通状態、すなわち、結合部730が規定

しているフォーマットエンジンの動作状態またはフォーマットエンジンが実行可能なアプリケーションの動作状態に基づいて指示を出す。この共通状態は、個別状態、すなわち、Javaミドルウェア740のアプリケーションマネージャ4503が実行するJavaアプリケーションの動作状態とは異なっている。この共通状態と個別状態との間のギャップを第1の変換部741が埋めるのである。第1の変換部741は、ナビゲータ720からの指示を、Javaミドルウェア740が取り扱う動作状態に変換する。その結果、アプリケーションマネージャ4503は、AITの制御情報やJavaアプリケーションからの指示と同等の指示を受け、Javaアプリケーションの動作状態を遷移させることができる。

第1の変換部741は、結合部730からのメッセージを変換してJavaミドルウェア740に伝え、また、Javaミドルウェア740の情報を変換して結合部730に伝えることにより、Javaミドルウェア740の動作を結合部730が規定する状態に適合させる。ここまで、様々なメッセージを、結合部730を通してJavaミドルウェア740に配達すると記述してきたが、正確には、これらのメッセージは、Javaミドルウェア740ではなく第1の変換部741に送信される。第1の変換部741が結合部730から「フォーマットエンジン状態要求」メッセージを受け取った場合、第1の変換部741は、Javaミドルウェア740が実行可能な全てのJavaアプリケーションの状態に基づいてJavaミドルウェア7

40の状態を決定する。そして、第1の変換部741は、「フォーマットエンジン状態応答」メッセージを作成し、結合部730へ送る。

図49は、Javaミドルウェア740が実行可能な全てのJavaアプリケーションの動作状態と、Javaミドルウェア740の動作状態との対応を示す変換表の一例を示す図である。図49において左側の列には、Javaミドルウェア740が実行可能な全てのJavaアプリケーションの動作状態が記述されている。右側の列には、対応するJavaミドルウェア740の動作状態が記述されている。ここで、Javaミドルウェア740の状態は、結合部730が規定した状態、すなわち共通状態である。具体的には、図21で示した「実行中」、「一時停止中」、および「停止中」という3つの動作状態である。上から2番目から4番目の行には、これら3つの動作状態に対応する、Javaミドルウェア740が実行可能な全てのJavaアプリケーションの動作状態が規定されている。

まず、上から2番目の行には、全てのJavaアプリケーションの中に1つでも「Active」という動作状態のJavaアプリケーションがあることに対しては、Javaミドルウェア740の動作状態として「実行中」が対応付けられている。一般に、「Active」状態のJavaアプリケーションは極小リソースを使用している可能性が高いため、このような対応関係を規定したものである。

次に、上から3番目の行には、「Active」状態の

J a v a アプリケーションがなく、かつ、全 J a v a アプリケーションの中で 1 つでも「P a u s e d」状態の J a v a アプリケーションがあることに対して、J a v a ミドルウェア 7 4 0 の動作状態として「一時停止中」が対応付けられている。D V B - M H P 1 . 0 規格によれば、「P a u s e d」状態の J a v a アプリケーションは、必要最小限のリソース以外は解放すべきと記述されている。よって、このような対応関係を規定した。

次に、上から 4 番目の行には、上から 2 番目および 3 番目の行に示す場合以外の場合に対して、「停止中」が対応付けられている。上から 2 番目および 3 番目の行に示す場合以外の場合は、具体的には、全ての J a v a アプリケーションが、「L o a d e d」あるいは「D e s t r o y e d」という動作状態、あるいは読み込みも完了していない動作状態にある場合である。この場合、J a v a アプリケーションのコードは 1 つも実行されていないので、リソースを一切使用していない「停止中」に対応付けるのが妥当である。

前述した通り、J a v a ミドルウェア 7 4 0 のアプリケーションマネージャ 4 5 0 3 は、A I T の制御情報の変化や J a v a アプリケーションの指示により、J a v a アプリケーションの状態を遷移させる。第 1 の変換部 7 4 1 は、状態変化の通知をアプリケーションマネージャ 4 5 0 3 から受ける。その結果として J a v a ミドルウェアの共通状態が変化する場合には、第 1 の変換部 7 4 1 は、図 4 9 の変換表を参照して、「フォーマットエンジン状態変化」

メッセージを生成して、結合部 730 に通知する。この場合、第 1 の変換部 741 は、1 次記憶部 109 に Java ミドルウェアの共通状態を保存しておく。そして、第 1 の変換部 741 は、Java アプリケーションの状態が変化した後の Java ミドルウェアの共通状態と比較する。なお、第 1 の変換部 741 は、Java アプリケーションの状態の変化をアプリケーションマネージャ 4503 から通知されたとき、常に「フォーマットエンジン状態変化」メッセージを生成して、結合部 730 に通知してもよい。

第 1 の変換部 741 が結合部 730 から「フォーマットエンジン実行」メッセージを受け取った場合、第 1 の変換部 741 は、AIT で規定された所定の Java アプリケーションを「Active」状態に遷移させるようにアプリケーションマネージャ 4503 に指示する。ここで所定の Java アプリケーションとは、AIT に規定されている全ての Java アプリケーションであってもよいし、優先度が最も高い Java アプリケーションであってもよい。所定の Java アプリケーションを「Active」状態に遷移させるには、DVB-MHP 1.0 規格が規定している org.dvb.application パッケージの機能を利用することができる。状態を変化させるべき Java アプリケーションの ID を指定した後、遷移させる状態「Active」を指定することによって、対象の Java アプリケーションの状態を「Active」状態に遷移させることができる。なお、すでに「Active」状態の Java アプリケーションが存在する場合、第 1

の変換部 741 は何もしないこととしてもよい。

第1の変換部 741 が結合部 730 から「フォーマットエンジン停止」メッセージを受け取った場合、第1の変換部 741 は、AIT に規定された全ての Java アプリケーションを「Destroyed」状態に遷移させるよう アプリケーションマネージャ 4503 に指示する。ただし、読み込んでいない Java アプリケーションに対しては何もしなくてもよい。全ての Java アプリケーションを「Destroyed」状態に遷移させるには、DVB-MHP 1.0 規格が規定している org.dvb.application パッケージの機能を利用することができる。Java アプリケーションの ID を指定した後、遷移させる状態「Destroyed」を指定することによって、対象の Java アプリケーションの状態を「Destroyed」状態に遷移させることができる。この操作をAIT が定義する全 Java アプリケーションに対して行えばよい。

第1の変換部 741 が結合部 730 から「フォーマットエンジン一時停止」メッセージを受け取った場合、第1の変換部 741 は、AIT に規定された Java アプリケーションの中で「Active」状態の Java アプリケーションを見つけ、その Java アプリケーションの状態を「Paused」状態に遷移させるようにアプリケーションマネージャ 4503 に指示する。この処理を実現するには、DVB-MHP 1.0 規格が規定している org.dvb.application パッケージの機能を利用す

ることができる。JavaアプリケーションのIDを指定した後、そのJavaアプリケーションの状態を取得する。もし状態が「Active」であれば、遷移させる状態「Paused」を指定することによって、対象のJavaアプリケーションの状態を「Paused」状態に遷移させることができる。この操作をAITが定義する全てのJavaアプリケーションに対して行えばよい。

第1の変換部741が結合部730から「アプリケーション・データ一覧要求」メッセージを受け取った場合、第1の変換部741は、AITが定義している全てのJavaアプリケーションおよびその状態に基づいて、「アプリケーション・データ一覧応答」メッセージを作成し、結合部730に送る。「アプリケーション・データ一覧応答」メッセージに含めるアプリケーション名は、AITが規定するアプリケーション名がそのまま使用される。「アプリケーション・データ一覧応答」メッセージに含める状態IDは、各Javaアプリケーションの状態を結合部730が規定する共通状態に変換表を用いて変換することによって作成される。

図50は、各Javaアプリケーションの動作状態を共通状態に変換するための変換表の一例を示す図である。図50において左側の列には、個別状態、すなわち、DVB-MHP1.0規格が規定しているJavaアプリケーションの動作状態が記述されている。右側の列には、対応する共通状態が記述されている。上から2番目の行に示されるように、DVB-MHP1.0規格が規定しているJa

v a アプリケーションの動作状態「Active」は、「実行中」という共通状態に対応付けられている。一般に、「Active」状態の Java アプリケーションは極小リソースを使用している可能性が高いので、このような対応関係を規定した。

また、図 50において上から 3 番目の行に示されるように、「DVB-MHP 1.0 規格が規定している Java アプリケーションの動作状態「Paused」は、「一時停止中」という共通状態に対応付けられている。DVB-MHP 1.0 規格によれば、「Paused」状態の Java アプリケーションは、必要最小限のリソース以外は解放すべきと記述されている。よって、このような対応関係を規定した。

また、上から 4 番目の行に示されるように、上から 2 番目および 3 番目の行以外の場合の Java アプリケーションの動作状態に対しては、「停止中」という共通状態が対応付けられている。上から 2 番目および 3 番目の行以外の場合とは、具体的には、Java アプリケーションの動作状態が、Java アプリケーションが「Loaded」もしくは「Destroyed」状態、または読み込みも完了していない状態にある場合である。この場合、Java アプリケーションのコードは 1 つも実行されていないので、リソースを一切使用していない「停止中」に対応付けるのが妥当である。

今、アプリケーション名が「EPG」である Java アプリケーションの動作状態が「Active」であり、ア

プリケーション名が「トランプゲーム」である Java アプリケーションの動作状態が「D e s t r o y e d」であるように、AITにおいて定義されている場合を考える。この場合、第1の変換部 741 は、「アプリケーション・データ一覧要求」メッセージを受け取ると、図 51 に示す「アプリケーション・データ一覧応答」メッセージ 5100 を生成し、結合部 730 に送信する。

図 51 は、第1の変換部 741 が送信するメッセージの具体例を示す図である。図 51 に示すメッセージ 5100 は、Java ミドルウェア 740 からナビゲータ 720 への「アプリケーション・データ取得応答」である。Source ID フィールド 5101 には、Java ミドルウェア 740 を表すサブプログラム ID として「2」が格納される（図 18 参照）。Destination ID フィールド 5102 には、ナビゲータ 720 を表すサブプログラム ID として「1」が格納される（図 18 参照）。Message ID フィールド 5103 には、「アプリケーション・データ取得応答」を表すメッセージ ID として「12」が格納される（図 19 参照）。Data Length フィールド 5104 には、データの長さを示す「27」が格納される。アプリケーション数フィールド 5105 には、Java ミドルウェア 740 が受け取ったAIT に規定されている Java アプリケーション数である「2」が格納される。

また、図 51において、アプリケーション数フィールド 5105 には、1 つ目の Java アプリケーションに対する

るアプリケーション情報フィールド 5_1_1_1 と、2 つ目の Java アプリケーションに対するアプリケーション情報フィールド 5_1_1_2 とが含まれている。アプリケーション I D フィールド 5_1_2_1 には、1 つ目の Java アプリケーションを表すアプリケーション I D として「1」が格納されている。アプリケーション状態 I D フィールド 5_1_2_2 には、「実行中」を表すアプリケーション状態 I D として「3」が格納されている。この「実行中」は、Java アプリケーションの動作状態「Active」が変換された結果である。アプリケーション名長さフィールド 5_1_2_3 には、アプリケーション名の長さを示す「6」が格納される。アプリケーション名フィールド 5_1_2_4 には、A I T で定義されたアプリケーション名「EPG」が格納されている。ここで「EPG」の各文字は 2 バイトコードで表現されており、結果、アプリケーション名長さは、3 文字 × 2 バイト = 6 バイトとなっている。アプリケーション I D フィールド 5_1_2_5 には 2 つ目の Java アプリケーションを表すアプリケーション I D として「2」が格納されている。状態 I D フィールド 5_1_2_6 には、「停止中」を表すアプリケーション状態 I D として「3」が格納されている。この「停止中」は、Java アプリケーションの動作状態「Destroyed」が変換された結果である。アプリケーション名長さフィールド 5_1_2_7 には、アプリケーション名の長さを示す「14」が格納される。アプリケーション名フィールド 5_1_2_8 には、A I T で定義されたアプリケーション名「トランプゲーム」が格納されてい

る。ここで「トランプゲーム」の各文字は2バイトコードで表現されており、結果、アプリケーション名長さは、7文字×2バイト=14バイトとなっている。

前述した通り、Javaミドルウェア740のアプリケーションマネージャ4503は、AITの制御情報の変化やJavaアプリケーションの指示によってJavaアプリケーションの動作状態を遷移させる。第1の変換部741は、状態変化の通知をアプリケーションマネージャ4503より受ける。その時、第1の変換部741は、「アプリケーション・データ一覧変化」メッセージを生成して、結合部730に通知する。第1の変換部741が結合部730から「アプリケーション・データ実行」メッセージを受け取った場合、第1の変換部741は、メッセージのデータフィールド175で指定されるJavaアプリケーションの状態を「Active」状態に遷移させるようにアプリケーションマネージャ4503に指示する。Javaアプリケーションを「Active」状態に遷移させるには、DVB-MHP1.0規格が規定しているorg.dvb.applicationパッケージの機能を利用することができる。状態を変化させるJavaアプリケーションのIDを指定した後、遷移させる状態「Active」を指定することによって、対象のJavaアプリケーションの状態を「Active」状態に遷移させることができる。なお、すでにJavaアプリケーションが「Active」状態の場合、第1の変換部741は、何もしないようにしてもよい。

第1の変換部741が結合部730から「アプリケーション・データ停止」メッセージを受け取った場合、第1の変換部741は、メッセージのデータフィールド175で指定されるJavaアプリケーションの状態を「Destroyed」に遷移させるようにアプリケーションマネージャ4503に指示する。ただし、読み込んでいないJavaアプリケーションに対しては何もしなくてもよい。Javaアプリケーションを「Destroyed」状態に遷移させるには、DVB-MHP1.0規格が規定しているorg.dvb.applicationパッケージの機能を利用することができる。JavaアプリケーションのIDを指定した後、遷移させる状態「Destroyed」を指定することによって、対象のJavaアプリケーションの状態を「Destroyed」状態に遷移させることができる。

第1の変換部741が結合部730から「アプリケーション・データ一時停止」メッセージを受け取った場合、第1の変換部741は、メッセージのデータフィールド175で指定されるJavaアプリケーションの状態を「Paused」に遷移させるようにアプリケーションマネージャ4503に指示する。この処理を実現するには、DVB-MHP1.0規格が規定しているorg.dvb.applicationパッケージの機能を利用することができる。JavaアプリケーションのIDを指定した後、遷移させる状態「Paused」を指定することによって、対象のJavaアプリケーションの状態を「Paused

」状態に遷移させることができる。

さらに、第1の変換部741は、結合部730のリソース管理部733からリソース剥奪の通知を受けてもよい。第1の変換部741は、剥奪されたリソースに応じて、アプリケーションマネージャ4503にJavaアプリケーションの状態を遷移させることとしてもよい。Javaアプリケーションの実行に不可欠なリソースが剥奪された場合、第1の変換部741は、全てのJavaアプリケーションの状態を「Destroyed」に遷移させる。それ以外のリソース、すなわち、Javaアプリケーションの実行に必ずしも必要ではないリソースが剥奪された場合については、第1の変換部741は、何も行わなくともよい。

また、リソース剥奪通知は、Javaミドルウェア740が直接受け取り、対応する処理を行ってもよい。以上で、Javaミドルウェア740の説明を終了する。

次に、図52～図71を用いて、HTMLブラウザ750について説明する。HTMLブラウザ750は、HTMLデータを解釈し表示を行うフォーマットエンジンである。本構成例においては具体的な例として、DVB-MHP1.1規格（正式には、ETSI TS 101 812 DVB-MHP仕様V1.1）が規定するDVB-HTMLを取り上げるが、例えば、 XHTML 1.0 等の他の仕様に従って動作するものでも実施可能である。本構成例では、HTMLブラウザ450の詳細な動作にはDVB-MHP1.1規格の内容が含まれる。従って、ここでは、

概略のみを記述する。

図52は、H T M L ブラウザ750の構成を示すプロック図である。H T M L ブラウザ750は、パーザー5201と、レイアウター5202と、描画部5203と、インターフェクション部5204と、H T M L マネージャ5205とを備えている。

パーザー5201は、H T M L マネージャ5205がダウンロードしたD V B - H T M L データの解析を行いD O M（ドキュメントオブジェクトモデル）ツリーを構築する。図53は、D V B - H T M L データの一例を示す図である。D V B - H T M L データに対して、パーザー5201は図55に示すようなD O Mツリーを構築する。

レイアウター5202は、パーザー5201が構築したD O Mツリーの各構成要素に対して、ディスプレイ107上の表示位置を計算して決定する。図54は、D V B - H T M L データのレイアウトを決めるスタイルシートの一例を示す図である。図54に示す各行は各々、図53に示すD V B - H T M L データ中の各要素5311, 5312, 5313, および5314の表示位置を指定している。ここでは、スタイルシートの例としてD V B - M H P が規定するC S S (W 3 C が規定するC S S 2 の拡張仕様)を取り上げているが、レイアウトを決めようとするH T M L データのスタイルシートとしてレイアウターが解析可能なものであれば、C S S 2 等の他の仕様に従うスタイルシートであっても実施可能である。レイアウター5202は、スタイルシートに基づいて、D O Mツリーの各構成要素に対

してディスプレイ 107 上の表示位置を決定する。例えば、図 55 に示す “Weather forecast” に対して (50, 50) で示される表示位置が決定された場合、文字列 “Weather forecast” は、その左上の座標がディスプレイの上から 50px, 左から 50px の位置となるように表示される。図 55 に示す “Japan” に対して (200, 100) で示される表示位置が決定された場合、文字列 “Japan” は、その左上の座標が、ディスプレイの上から 200px, 左から 100px の位置となるように表示される。

なお、レイアウターがスタイルシートのパーザーを持ち、DOMツリー構築後においてそのパーザーが HTML マネージャから指示を受けることによってスタイルシートを解析・計算し、計算した結果の表示位置をレイアウターに通知してもよい。また、HTML ブラウザがデフォルトのスタイルシートを持ち、スタイルシートが存在しない場合は、レイアウターは、HTML ブラウザが持つデフォルトのスタイルシートに従い、DOMツリーの各構成要素に対して、ディスプレイ 107 上の表示位置を計算して決定してもよい。

図 52 の説明に戻り、描画部 5203 は、レイアウター 5202 が計算した表示位置に従って、DOMツリーの各構成要素を描画する。図 56 は、図 55 に示す DOMツリーを描画した際のディスプレイ 107 を示す図である。

インターフェクション部 5204 は、描画した DVB-HTML データに含まれるリンクやボタン等に対してユーザが

入力した指示に対応する処理を行う。例えば、DVB-H HTMLデータ内に他のDVB-H HTMLデータへのリンクが定義されているとする。ここで、ユーザが、リンクのクリックを入力部111から入力すると、インタラクション部5204は、リンクに定義されたDVB-H HTMLデータの情報をHTMLマネージャ5205に引き渡す。

HTMLマネージャ5205は、引き渡されたDVB-H HTMLデータの情報に基づいて、DVB-H HTMLデータをダウンロードする。そして、HTMLマネージャ5205は、パーザー5201、レイアウター5202、および描画部5203を用いてDVB-H HTMLデータを解釈・表示する。図53に示すDVB-H HTMLデータ内には、他のDVB-H HTMLデータへのリンクが定義されている。

図56において、ディスプレイ上の文字列“Japan”として代表されるリンク先「d v b : // 1. 2. 1 / Japan_Weather.html」のクリックをユーザが入力すると、インタラクション部5204は、DVB-H HTMLデータの情報「d v b : // 1. 2. 1 / Japan_Weather.html」をHTMLマネージャ5205に引き渡す。ここで、あるリンク先のクリックを入力するとは、そのリンク先をフォーカス5601が選択しているときにOKボタン305を押下することをいう。HTMLマネージャ5205は、DVB-H HTMLデータの情報「d v b : // 1. 2. 1 / Japan_Weather.html」に基づいて、DVB-H HTML

データをダウンロードする。そして、H T M L マネージャ 5 2 0 5 は、パーザー 5 2 0 1 、レイアウター 5 2 0 2 、描画部 5 2 0 3 を用いて D V B - H T M L データを解釈・表示する。図 5 7 は、ユーザが文字列 “ J a p a n ” と代表されるリンク先をクリックした後にディスプレイ表示される D V B - H T M L データの例を示す図である。

また、例えば、D V B - H T M L データ内に定義されたボタンに、D O M ツリーの構成を変更するスクリプトが定義されているとする。ここで、ユーザがボタンのクリックを入力部 1 1 1 から入力すると、インターラクション部 5 2 0 4 は、パーザー 5 2 0 1 から D V B - H T M L データを受け取り、ボタンに定義されたスクリプトを実行し、D O M ツリーの構成を変更する。この変更に伴い、レイアウター 5 2 0 2 が表示位置を再計算し、描画部 5 2 0 3 は描画をやり直す。また、D O M ツリーを変更したことを、H T M L マネージャ 5 2 0 5 に通知する。

図 5 8 に示す D V B - H T M L データ内には、D O M ツリーの構成を変更するスクリプト 5 8 0 1 として “ c h a n g e D O M T r e e () ” (具体的には、例えば、D O M ツリーにおいてテーブル要素を記述したもの) が定義され、ボタン 5 8 0 2 のクリック時にそのスクリプト 5 8 0 1 を実行するよう定義されている。スクリプト 5 8 0 1 において、D O M ツリーの構成を変更するソースコードは省略されている。スクリプトは、例えば、D V B - M H P が規定する D O M (W 3 C が規定する D O M 2 の拡張仕様) 仕様に従うものであっても、D O M 1 、 2 等の他の仕様に

従うスクリプトであっても実施可能である。

図59は、図58に示すDVB-HHTMLデータに対してパーザー5201が構築したDOMツリーの一部を示す図である。また、図60は、レイアウト5202および描画部5203によって表示されたディスプレイ表示の例を示す図である。図60において、ラベル“詳細情報”6001に代表されるボタン5802のクリックをユーザが入力すると、インタラクション部5204は、パーザー5201からDVB-HHTMLデータを受け取る。そして、インタラクション部5204は、ボタン5802に定義されたスクリプト5801を実行してDOMツリーの構成を変更する。また、レイアウト5202は、表示位置を最計算する。描画部5203は、描画をやり直す。さらに、DOMツリーを変更したことがHTMLマネージャ5205に通知される。図61は、ユーザによるボタンのクリックに伴うDOMツリーの変更結果を示す図である。また、図62は、ディスプレイ表示の変更結果の例を示す図である。

HTMLマネージャ5205は、放送波中に多重化されているDVB-HHTMLデータをダウンロードする。そして、HTMLマネージャ5205は、パーザー5201、レイアウト5202、および描画部5203を用いてDVB-HHTMLデータを解釈し、表示する。MHP1.1規格によれば、放送波中には、AITと呼ばれるDVB-HHTMLデータを定義する表が多重化され伝送される。HTMLマネージャ5205は、このAITを最初にダウン

ロードする。AITには、DVB-HHTMLデータをダウンロードするための情報や、DVB-HHTMLデータの名前、DVB-HHTMLデータの制御情報などが含まれる。

図63は、AITの主要部を表した模式図である。図63において最も左側の列には、DVB-HHTMLデータのIDが記述されている。左から2番目の列には、DVB-HHTMLデータの制御情報が記述されている。制御情報には「auto_start」、「present」、「kill」等がある。「auto_start」は、HTMLブラウザ750がDVB-HHTMLデータを即時に解釈・表示することを意味する。「present」は、自動的に解釈・表示しないことを意味する。「kill」は、DVB-HHTMLデータの表示を消すことを意味する。左から3番目の列には、DVB-HHTMLデータのデータ名が記述されている。最も右側の列には、DVB-HHTMLデータの優先度が記述されている。複数のDVB-HHTMLデータの表示時に、リソースの競合が発生した場合、優先度の高いDVB-HHTMLデータの表示にリソースが優先的に割り当てられる。ここで、優先度の値が大きいほど優先度が高いこととする。

図63において、上から2番目および3番目の行には、DVB-HHTMLデータの情報の組が記述される。上から2番目の行において定義されるDVB-HHTMLデータについては、IDが「1」であり、制御情報が「auto_start」であり、データ名が「天気予報」であり、優先度が「54」である。上から3番目の行において定義され

るDVB-HTM Lデータについては、IDが「2」であり、制御情報が「present」であり、データ名が「レジャー情報」であり、優先度が「22」である。

HTM Lマネージャ5205は、図63に示すAITを受け取ると、データ名が「天気予報」であるDVB-HTM Lデータをダウンロードし、解釈・表示する。ここで、DVB-MHP1.1規格が規定するDVB-HTM Lデータは、「Loaded」、「Paused」、「Active」、「Destroyed」、および「Killed」という5つの状態を有している。

図64は、DVB-HTM Lデータの5つの状態および状態間の遷移を表す状態遷移図である。HTM Lマネージャ5205がDVB-HTM Lデータをダウンロードし、1次記憶部109に格納されると、当該DVB-HTM Lデータの状態は「Loaded」になる。各状態間を結ぶ矢印は、状態間で可能な遷移を表しており、矢印のない状態間の遷移は発生しない。例えば、「Destroyed」の状態から「Active」の状態に遷移することはない。

HTM Lマネージャ5205は、DVB-HTM Lデータの状態遷移を、(1)放送波中のAITの制御情報、(2)DVB-HTM Lデータの指示、および(3)ナビゲータ720が結合部730を通して送る指示、の3通りの指示に従って行う。

AITの制御情報が「auto start」の場合、HTM Lマネージャ5205は、DVB-HTM Lデータを

ダウンロードした後、パーザー 3602、レイアウター 3603、および描画部 3604 に指示することで DVB-H HTML データを解釈・表示し、「Active」状態に遷移させる。AIT は、時間と共に変化する。例えば、AIT が図 63 に示す状態から図 65 に示す状態へ変化したとする。図 65 では、データ名が「天気予報」である DVB-H HTML データの制御情報が「auto_start」から「kill」に変更されている。このとき、HTML マネージャ 5205 は、DVB-H HTML データの表示状態を「Killed」という状態に遷移させる。

一方、DVB-H HTML データに対するユーザの入力により、DVB-H HTML データが状態遷移を発生させることができる。図 66 は、DVB-H HTML データ内に他の DVB-H HTML データへのリンクが定義されている例を示す図である。図 67 は、図 66 に示す DVB-H HTML データの表示例を示す図である。図 67において、表示 6711 がリンク 6601 に対応する。ここで、ユーザがリンク 6601 に対してクリックを入力部 111 から入力すると、インタラクション部 5204 は、リンクに定義された DVB-H HTML データの情報を HTML マネージャ 5205 に引き渡す。ここで、DVB-H HTML データの情報は図 66 を参照して「d v b : // 1. 2. 1 / J a p a n _ W e a t h e r. h t m l」である。HTML マネージャ 5205 は、引き渡された DVB-H HTML データの情報に基づいて、DVB-H HTML データをダウンロードする。そして、HTML マネージャ 5205 は、パーザ

— 5 2 0 1 、レイアウター 5 2 0 2 、および描画部 5 2 0 3 を用いて D V B - H T M L データを解釈・表示する。この処理によって、 H T M L マネージャは、元の D V B - H T M L データの状態を「 A c t i v e 」から「 K i l l e d 」へと遷移することになる。一方、「 d v b : // 1 . 2 . 1 / J a p a n _ W e a t h e r . h t m l 」という情報に相当する新たな D V B - H T M L データの状態は、「 L o a d i n g 」から「 A c t i v e 」へと遷移することになる。図 6 8 は、情報「 d v b : // 1 . 2 . 1 / J a p a n _ W e a t h e r . h t m l 」に相当する新たな D V B - H T M L データの表示例を示す図である。

最後に、ナビゲータ 7 2 0 からの指示によって D V B - H T M L データの表示の状態を遷移する方法を説明する。ナビゲータ 7 2 0 は、共通状態、すなわち、結合部 7 3 0 が規定しているフォーマットエンジンの状態およびフォーマットエンジンが表示可能なデータの状態に基づいて指示を出す。この共通状態は、個別状態、すなわち、 H T M L ブラウザ 7 5 0 の H T M L マネージャ 5 2 0 5 が表示する D V B - H T M L データの表示の状態および状態遷移とは異なる。この共通状態と個別状態との間のギャップを第 2 の変換部 7 5 1 が埋めるのである。第 2 の変換部 7 5 1 は、ナビゲータ 7 2 0 からの指示を、 H T M L ブラウザ 7 5 0 が取り扱う状態および状態遷移に変換する。その結果、 H T M L マネージャ 5 2 0 5 は、 A I T の制御情報や D V B - H T M L データからの指示と同等の指示を受け、 D V B - H T M L データの表示の状態を遷移させること

ができる。

第2の変換部751は、結合部730からのメッセージを変換してHTMLブラウザ750に伝え、また、HTMLブラウザ750の情報を変換して結合部730に伝えることにより、HTMLブラウザ750の動作を結合部730が規定する状態に適合させる。ここまで、様々なメッセージを、結合部730を通してHTMLブラウザ750に配達すると記述してきたが、正確には、これらのメッセージはHTMLブラウザ750ではなく第2の変換部751に送信される。第2の変換部751が結合部730から「フォーマットエンジン状態要求」メッセージを受け取った場合、第2の変換部751は、HTMLブラウザ750が表示可能な全てのHTMLデータの状態に基づいてHTMLブラウザ750の状態を決定する。そして、第2の変換部751は、「フォーマットエンジン状態応答」メッセージを作成し、結合部730へ送る。

図69は、HTMLブラウザ750が表示可能な全てのHTMLデータの状態と、HTMLブラウザ750の動作状態との対応を示す変換表の一例を示す図である。図69において左側の列には、HTMLブラウザ750が表示可能な全てのHTMLデータの状態が記述されている。右側の列には、対応するHTMLブラウザ750の動作状態が記述されている。ここで、HTMLブラウザ750の状態は、結合部730が規定した状態、すなわち共通状態である。具体的には、図21で示した「実行中」、「一時停止中」、および「停止中」という3つの動作状態である。上

から 2 番目から 4 番目の行には、これら 3 つの状態に対応する、H T M L ブラウザ 750 が表示可能な全ての H T M L データの状態が規定されている。

まず、上から 2 番目の行には、全ての H T M L データの中で 1 つでも「A c t i v e」状態の H T M L データがあることに対して、H T M L ブラウザ 750 の共通状態として「実行中」が対応付けられている。一般に、「A c t i v e」状態の H T M L データの表示およびユーザとのインタラクションには極小リソースを使用している可能性が高いため、このような対応関係を規定した。

次に、上から 3 番目の行には、「A c t i v e」状態の H T M L データがなく、かつ、全 H T M L データの中で 1 つでも「P a u s e d」状態の H T M L データがあることに対して、H T M L ブラウザ 750 の共通状態として「一時停止中」が対応付けられている。D V B - M H P 1.1 規格によれば、「P a u s e d」状態の H T M L データの表示は、一部のリソースは使用できないと記述されている。よって、このような対応関係を規定した。

次に、上から 4 番目の行には、上から 2 番目および 3 番目の行に示す場合以外の場合に対して、「停止中」が対応付けられている。上から 2 番目および 3 番目の行に示す場合以外の場合とは、具体的には、全ての H T M L データが「L o a d e d」、「D e s t r o y e d」、もしくは「K i l l e d」状態、または読み込みも完了していない状態にある場合である。この場合、H T M L データに対するインタラクション処理は実行されていないので、リソース

を一切使用していない「停止中」に対応付けるのが妥当である。

前述した通り、H T M L ブラウザ 7 5 0 の H T M L マネージャ 5 2 0 5 は、A I T の制御情報の変化や H T M L データの指示により、H T M L データの状態を遷移させる。第 2 の変換部 7 5 1 は、状態変化の通知を H T M L マネージャ 5 2 0 5 より受ける。その結果として H T M L ブラウザ 7 5 0 の状態が変化する場合には、第 2 の変換部 7 5 1 は、図 6 9 の変換表を参照して、「フォーマットエンジン状態変化」メッセージを生成して、結合部 7 3 0 に通知する。この場合、第 2 の変換部 7 5 1 は、1 次記憶部 1 0 9 に H T M L ブラウザ 7 5 0 の共通状態を保存しておく。そして、H T M L データの状態が変化した後の共通状態と比較する。なお、第 2 の変換部 7 5 1 は、H T M L データの状態の変化を H T M L マネージャ 5 2 0 5 から通知されたとき、常に「フォーマットエンジン状態変化」メッセージを生成して、結合部 7 3 0 に通知してもよい。

第 2 の変換部 7 5 1 が結合部 7 3 0 から「フォーマットエンジン実行」メッセージを受け取った場合、第 2 の変換部 7 5 1 は、A I T に規定された所定の H T M L データを「A c t i v e」状態に遷移させるように H T M L ブラウザ 7 5 0 に指示する。ここで所定の H T M L データとは、A I T に規定されている全ての H T M L データでもよいし、優先度が最も高い H T M L データであってもよい。所定の H T M L データを「A c t i v e」状態に遷移させるには、D V B - M H P 1 . 1 規格が規定している o r g . d

v b . a p p l i c a i t o n パッケージの機能を利用することができる。状態を変化させるべき H T M L データの I D を指定した後、遷移させる状態「A c t i v e 」を指定することによって、対象の H T M L データの状態を「A c t i v e 」状態に遷移させることができる。なお、すでに「A c t i v e 」状態の H T M L データが存在する場合は、第 2 の変換部 7 5 1 は何もしないということとしてもよい。

第 2 の変換部 7 5 1 が結合部 7 3 0 から「フォーマットエンジン停止」メッセージを受け取った場合、第 2 の変換部 7 5 1 は、A I T に規定された全ての H T M L データを「K i l l e d 」状態に遷移させるように H T M L ブラウザ 7 5 0 に指示する。ただし、読み込んでいない H T M L データに対しては何もしなくてよい。全ての H T M L データを「K i l l e d 」状態に遷移させるには、D V B - M H P 1 . 1 規格が規定している o r g . d v b . a p p l i c a i t o n パッケージの機能を利用することができる。H T M L データの I D を指定した後、遷移させる状態「K i l l e d 」を指定することで、対象の H T M L データの状態を「K i l l e d 」状態に遷移させることができる。この操作を A I T が定義する全 H T M L データに対して行えばよい。

第 2 の変換部 7 5 1 が結合部 7 3 0 から「フォーマットエンジン一時停止」メッセージを受け取った場合、第 2 の変換部 7 5 1 は、A I T に規定された H T M L データの中で「A c t i v e 」状態の H T M L データを見つけ、その

H T M L データの状態を「P a u s e d」状態に遷移させるようにH T M L ブラウザ750に指示する。この処理を実現するには、D V B - M H P 1. 1 規格が規定しているorg. d v b. a p p l i c a i t o n パッケージの機能を利用することができる。H T M L データのI D を指定した後、そのH T M L データの状態を取得する。もし状態が「A c t i v e」であれば、遷移させる状態「P a u s e d」を指定することによって、対象のH T M L データの状態を「P a u s e d」状態に遷移させることができる。この操作をA I T が定義する全てのH T M L データに対して行えばよい。

第2の変換部751が結合部730から「アプリケーション・データ一覧要求」メッセージを受け取った場合、第2の変換部751は、A I T が定義している全てのH T M L データおよびその状態に基づいて、「アプリケーション・データ一覧応答」メッセージを作成し、結合部730に送る。「アプリケーション・データ一覧応答」メッセージに含めるアプリケーション名は、A I T が規定するアプリケーション名がそのまま使用される。「アプリケーション・データ一覧応答」メッセージに含める状態I D は、各H T M L データの状態を結合部730が規定する共通状態に変換表を用いて変換することによって作成される。

図70は、各H T M L データの状態を共通状態に変換するための変換表の一例を示す図である。図70において左側の列には、D V B - M H P 1. 1 規格が規定しているH T M L データの状態が記述されている。右側の列には、対

応する共通状態が記述されている。上から 2 番目の行に示されるように、DVB-MHP1.1 規格が規定している HTML データの状態「Active」は、「実行中」という共通状態に対応付けられている。一般に、「Active」状態の HTML データの表示およびユーザとのインタラクションは極小リソースを使用する可能性が高いので、このような対応関係を規定した。

また、図 70において上から 3 番目の行に示されるように、DVB-MHP1.1 規格が規定している HTML データの状態「Paused」は、「一時停止中」という共通状態に対応付けられている。DVB-MHP1.1 規格によれば、「Paused」状態の HTML データの表示およびユーザとのインタラクション処理は使用可能なりソースが制限されると記述されている。よって、このような対応関係を規定した。

また、上から 4 番目の行に示されるように、上から 2 番目および 3 番目の行以外の場合の HTML データの状態に対しては、「停止中」という共通状態が対応付けられている。上から 2 番目および 3 番目の行以外の場合とは、具体的には、HTML データが「Loaded」、「Destroyed」、もしくは「Killed」状態、または読み込みも完了していない状態にある場合である。この場合、HTML データの解釈・表示処理およびユーザとのインタラクション処理は実行されないので、リソースを一切使用していない「停止中」に対応付けるのが妥当である。

今、データ名が「天気予報」である HTML データの状

態が「Active」であり、データ名が「レジャー情報」であるHTMLデータの状態が「Destroyed」であるように、AITにおいて定義されている場合を考える。この場合、第2の変換部751は、「アプリケーション・データ一覧要求」メッセージを受け取ると、図71に示す「アプリケーション・データ一覧応答」メッセージ470.0を生成し、結合部730に送信する。

図71は、第2の変換部751が送信するメッセージの具体例を示す図である。図71に示すメッセージ710.0は、HTMLブラウザ750からナビゲータ720への「アプリケーション・データ取得応答」である。Source IDフィールド710.1には、HTMLブラウザ750を表すサブプログラムIDとして「3」が格納される（図18参照）。Destination IDフィールド710.2には、ナビゲータ720を表すサブプログラムIDとして「1」が格納される（図18参照）。Message IDフィールド710.3には、「アプリケーション・データ取得応答」を表すメッセージIDとして「12」が格納される（図19参照）。Data Lengthフィールド710.4には、データの長さを表す「27」が格納される。アプリケーション数フィールド710.5には、HTMLブラウザ750が受け取ったAITに規定されているHTMLデータ数である「2」が格納される。

また、図71において、アプリケーション数フィールド710.5には、1つ目のHTMLデータに対するアプリケーション情報フィールド711.1と、2つ目のHTMLデ

ータに対するアプリケーション情報フィールド 7112 とが含まれている。アプリケーション ID フィールド 7121 には、1 つ目の H T M L データを表すアプリケーション ID として「1」が格納されている。アプリケーション状態 ID フィールド 7122 には、「実行中」を表すアプリケーション状態 ID として「3」が格納されている。この「実行中」は、H T M L データの状態「A c t i v e」が変換された結果である。アプリケーション名長さフィールド 7123 には、アプリケーション名の長さを示す「8」が格納される。アプリケーション名フィールド 7124 には、A I T で定義されたデータ名「天気予報」が格納されている。ここで「天気予報」の各文字は 2 バイトコードで表現されており、結果、アプリケーション名長さは、4 文字 × 2 バイト = 8 バイトとなっている。アプリケーション ID フィールド 7125 には 2 つ目の H T M L データを表すアプリケーション ID として「2」が格納されている。状態 ID フィールド 7126 には、「停止中」を表すアプリケーション状態 ID として「3」が格納されている。この「停止中」は、J a v a アプリケーションの動作状態「D e s t r o y e d」が変換された結果である。アプリケーション名長さフィールド 7127 には、アプリケーション名の長さを示す「12」が格納される。アプリケーション名フィールド 7128 には、A I T で定義されたデータ名「レジヤー情報」が格納されている。ここで「レジヤー情報」の各文字は 2 バイトコードで表現されており、結果、アプリケーション名長さは、6 文字 × 2 バイト = 12 バ

イトとなっている。

前述した通り、H T M L ブラウザ 7 5 0 の H T M L マネージャ 5 2 0 5 は、A I T の制御情報の変化や H T M L データの指示によって、H T M L データの状態を遷移させる。第 2 の変換部 7 5 1 は、状態変化の通知を H T M L マネージャ 5 2 0 5 より受ける。その時、第 2 の変換部 7 5 1 は、「アプリケーション・データ一覧変化」メッセージを生成して、結合部 7 3 0 に通知する。第 2 の変換部 7 5 1 が、結合部 7 3 0 から「アプリケーション・データ実行」メッセージを受け取った場合、第 2 の変換部 7 5 1 は、メッセージのデータフィールド 1 7 5 で指定される H T M L データの状態を「A c t i v e」状態に遷移させるように H T M L マネージャ 5 2 0 5 に指示する。H T M L データを「A c t i v e」状態に遷移させるには、D V B - M H P 1 . 1 規格が規定している o r g . d v b . a p p l i c a i t o n パッケージの機能を利用することができる。状態を変化させる H T M L データの I D を指定した後、遷移させる状態「A c t i v e」を指定することによって、対象の H T M L データの状態を「A c t i v e」状態に遷移させることができる。なお、すでに H T M L データが「A c t i v e」状態の場合、第 2 の変換部 7 5 1 は何もしないとようしてもよい。

第 2 の変換部 7 5 1 が結合部 7 3 0 から「アプリケーション・データ停止」メッセージを受け取った場合、第 2 の変換部 7 5 1 は、メッセージのデータフィールド 1 7 5 で指定される H T M L データの状態を「K i l l e d」に遷

移させるように H T M L マネージャ 5 2 0 5 に指示する。ただし、読み込んでいない H T M L データに対しては何もしなくてもよい。H T M L データを「K i l l e d」状態に遷移させるには、D V B - M H P 1 . 1 規格が規定している o r g . d v b . a p p l i c a i t o n パッケージの機能を利用することができる。H T M L データの I D を指定した後、遷移させる状態「K i l l e d」を指定することで、対象の H T M L データの状態を「K i l l e d」状態に遷移させることができる。

第 2 の変換部 7 5 1 が結合部 7 3 0 から「アプリケーション・データ一時停止」メッセージを受け取った場合、第 2 の変換部 7 5 1 は、メッセージのデータフィールド 1 7 - 5 で指定される H T M L データの状態を「P a u s e d」に遷移させるように H T M L マネージャ 5 2 0 5 に指示する。この処理を実現するには、D V B - M H P 1 . 1 規格が規定している o r g . d v b . a p p l i c a i t o n パッケージの機能を利用することができる。H T M L データの I D を指定した後、遷移させる状態「P a u s e d」を指定することで、対象の H T M L データの状態を「P a u s e d」状態に遷移させることができる。

さらに、第 2 の変換部 7 5 1 は、結合部 7 3 0 のリソース管理部 7 3 3 よりリソース剥奪の通知を受けてもよい。第 2 の変換部 7 5 1 は、剥奪されたリソースに応じて、H T M L マネージャ 5 2 0 5 に H T M L データの状態を遷移させることとしてもよい。H T M L データの表示に不可欠なリソースが剥奪された場合は、全ての H T M L データの

状態を「K i l l e d」に遷移させる。それ以外のリースの剥奪に対しては、第2の変換部751は、何も行わないとしてもよい。

また、リソース剥奪通知は、H T M L ブラウザ750が直接受け取り、対応する処理を行ってもよい。以上で、H T M L ブラウザ750の説明を終了する。

次に、図72および図73を用いて、メーラー760について説明する。メーラー760は、電子メールの読み書きを行うためのフォーマットエンジンである。メーラー760は、J a v a ミドルウェア740やH T M L ブラウザ750のように、取り扱うアプリケーションやデータに関する状態を規定していない。メーラー760自身の状態も特に規定がない。ここでは、メーラー760は、実行中および停止中という2つの状態を有している考える。ここで、メーラー760は一般的な電子メールソフトウェアの機能を含む。メーラー760自体の機能は既知であるので説明を省略する。

第3の変換部761は、結合部730からのメッセージを変換してメーラー760に伝える。また、第3の変換部761は、メーラー760の動作状態を変換して結合部730に伝える。これによって、第3の変換部761は、メーラー760の動作状態を結合部730が規定する共通状態に適合させる。なお、ここまで、様々なメッセージを、結合部730を通してメーラー760に配達すると記述してきたが、正確には、これらのメッセージはメーラー760ではなく第3の変換部761に配達される。

第3の変換部761が結合部730から「フォーマットエンジン状態要求」メッセージを受け取った場合、第3の変換部761は、メーラー760の動作状態に基づいて「フォーマットエンジン状態応答」メッセージを作成し、結合部730に送る。

図72は、メーラー760の状態と、結合部730が規定している共通状態との対応を示す変換表の一例を示す図である。図72において左側の列には、メーラー760の個別状態が記述されている。右側の列には、対応するメーラー760の共通状態が記述されている。共通状態は、具体的には、図21で示した「実行中」および「停止中」という2つの動作状態である。上から2番目および3番目の行には、これら2つの状態に対応する、メーラー760の個別状態が規定されている。

まず、上から2番目の行には、メーラー760が起動中であることに対して、メーラー760の共通状態として「実行中」が対応付けられている。一般に、起動中のメーラーはネットワークインターフェース等の極小リソースを使用している可能性が高いため、このような対応関係を規定した。

次に、上から3番目の行には、メーラー760が停止中であることに対して、メーラー760の共通状態として「停止中」が対応付けられている。メーラー760が停止しているときは、メーラー760はリソースを一切使用していない。よって、「停止中」に対応付けるのが妥当である。なお、メーラー760には、共通状態「一時停止中」に

対応する個別状態はない。

メーラー760は、不測の事態などが発生すると、メーラー760自身を終了する可能性がある。第3の変換部761は、この変化をメーラー760より受ける。このとき、第3の変換部761は、「フォーマットエンジン状態変化」メッセージを生成して、結合部730に通知する。第3の変換部761が結合部730から「フォーマットエンジン実行」メッセージを受け取った場合、第3の変換部761は、メーラー760を起動する。第3の変換部761が結合部730から「フォーマットエンジン停止」メッセージを受け取った場合、第3の変換部761は、メーラー760を終了させる。第3の変換部761が結合部730から「フォーマットエンジン一時停止」メッセージを受け取った場合、第3の変換部761は、メーラー760を終了させる。図72で示すように、メーラー760には「一時停止中」に対応する状態がない。ここでは、「一時停止中」とは、極小リソースを使用しないという意味である（図21参照）。これを実現するには、メーラー760を終了させるしかない。

第3の変換部761が結合部730から「アプリケーション・データ一覧要求」メッセージを受け取った場合、「アプリケーション・データ一覧応答」メッセージを作成し、結合部730に送る。ここで、メーラー760は、取り扱うアプリケーションやデータがないため、アプリケーション数0のメッセージを生成する。

図73は、第3の変換部761が生成する「アプリケー

「ショット・データー覧応答」メッセージの具体例を示す図である。図73に示すメッセージ7300は、メーラー760からナビゲータ720への「アプリケーション・データ取得応答」である。Source IDフィールド7301には、メーラー760を表すサブプログラムIDとして「4」が格納される。Destination IDフィールド7302には、ナビゲータ720を表すサブプログラムIDとして「1」が格納される。Message IDフィールド7303には、「アプリケーション・データ取得応答」を表すメッセージIDとして「12」が格納される。Data Lengthフィールド7304には、データの長さを示す「1」が格納される。アプリケーション数フィールド7305には、「0」が格納される。

第3の変換部761が結合部730から「アプリケーション・データ実行」メッセージ、「アプリケーション・データ停止」メッセージあるいは「アプリケーション・データ時停止」メッセージを受け取った場合、第3の変換部761は、何もしない。この理由は、メーラー760は、取り扱うアプリケーション・データの状態を持たないからである。また、第3の変換部761は、「アプリケーション・データー覧変化」メッセージを生成することはない。

さらに第3の変換部761は、結合部730のリソース管理部733よりリソース剥奪の通知を受けてもよい。第3の変換部761は、剥奪されたリソースに応じて、メーラー760を終了させることとしてもよい。メーラーの実行に不可欠なリソース、例えばネットワークインターフェ

ース等が剥奪された場合は、第3の変換部761は、メーラー760を終了させる。それ以外のリースの剥奪に対しては、第3の変換部761は、何も行わないとしてもよい。

また、リソース剥奪通知は、メーラー760が直接受け取り、対応する処理を行ってもよい。以上で、メーラー760の説明を終了する。また、以上で、情報処理装置の具体的な構成の説明を終了する。

(詳細な動作)

次に、図74～図84を用いて、上記の構成例に示したデジタルテレビにおいて前述した第1および第2の動作例の動作を行う場合について説明する。

(第1の動作例の詳細な動作)

まず、図74～図81を用いて、第1の動作例の詳細な動作を説明する。図74は、第1の動作例の動作を行う場合におけるデジタルテレビにおける処理の流れを示すフローチャートである。図74において、まず、ナビゲータ720は、フォーマットエンジンの起動要求を受け付ける(ステップS1)。ここで、起動要求は、典型的にはユーザによって指示される。例えば図8に示す状態において、ユーザが入力部111のOKボタン305を押下することによって、Javaミドルウェア740に関する起動要求が指示される。

次に、ナビゲータ720は、各フォーマットエンジンの内、1つを選択する(ステップS2)。ステップS2で選択したフォーマットエンジンについて、以降のステップS

3～S 6 の処理が行われる。続いて、ナビゲータ 720 は、ステップ S 2 で選択したフォーマットエンジンに対する「フォーマットエンジン（図では、“F M”と記載する。）状態要求」メッセージを結合部 730 へ送信する（ステップ S 3）。このメッセージは、結合部 730 によって当該フォーマットエンジンに対応する変換部へ送信される。

ここで、結合部 730 の通信部 731 がメッセージを送信する処理について説明する。図 75 は、通信部 731 が行うメッセージ送信の処理の流れを示すフローチャートである。通信部 731 は、メッセージを受け取ると（ステップ S 7501）、メッセージ中の Destination ID フィールドの値を参照する。Destination ID フィールドの値が「0」のとき（ステップ S 7502）、通信部 731 は、ナビゲータ 720、Java ミドルウェア 740、HTML ブラウザ 775、メーラー 760、および結合部 730 の状態管理部 732 にメッセージを送信する（ステップ S 7503）。Destination ID フィールドの値が「1」のとき（ステップ S 7504）、通信部 731 は、ナビゲータ 720 にメッセージを送信する（ステップ S 7505）。Destination ID フィールドの値が「2」のとき（ステップ S 7506）、通信部 731 は、Java ミドルウェア 740 にメッセージを送信する（ステップ S 7507）。Destination ID フィールドの値が「3」のとき（ステップ S 7508）、通信部 731 は、HTML ブラウザ 775 にメッセージを送信する（ステップ S 7509）。

9)。Destination ID フィールドの値が「4」のとき（ステップ S 7510）、通信部 731 は、メーラー 760 にメッセージを送信する（ステップ S 7511）。Destination ID フィールドの値が「5」のとき（ステップ S 7512）、通信部 731 は、結合部 730 の状態管理部 732 にメッセージを送信する（ステップ S 7513）。以上の処理によって、各サブプログラムからのメッセージは、適切なサブプログラムへ送信されることとなる。通信部 731 は、メッセージを受け取る度に、上記処理を行う。

次に、ステップ S 3 の「フォーマットエンジン状態要求」メッセージを受信した際の各変換部の処理の流れについて説明する。

図 76 は、第 1 の変換部 741 が「フォーマットエンジン状態要求」メッセージを受け取ったときの処理の流れを示すフローチャートである。第 1 の変換部 741 が「フォーマットエンジン状態要求」メッセージを受け取ると（ステップ S 7601）、第 1 の変換部 741 は、「Active」状態の Java アプリケーションがあるかどうかを調べる（ステップ S 7602）。「Active」状態の Java アプリケーションがある場合、第 1 の変換部 741 は、Java ミドルウェア 740 が「実行中」であることを示す「フォーマットエンジン状態応答」メッセージを作成する（ステップ S 7603）。一方、「Active」状態の Java アプリケーションがない場合、第 1 の変換部 741 は、「Paused」状態の Java アプリケ

ーションがあるかどうかを調べる（ステップ S 7 6 0 4）。 「Paused」状態の Java アプリケーションがある場合、第 1 の変換部 7 4 1 は、Java ミドルウェア 7 4 0 が「一時停止中」であることを示す「フォーマットエンジン状態応答」メッセージを作成する（ステップ S 7 6 0 5）。一方、「Paused」状態の Java アプリケーションがない場合、第 1 の変換部 7 4 1 は、Java ミドルウェア 7 4 0 が「停止中」である「フォーマットエンジン状態応答」メッセージを作成する（ステップ S 7 6 0 6）。ステップ S 7 6 0 3、S 7 6 0 5、または S 7 6 0 6 の後、第 1 の変換部 7 4 1 は、作成した「フォーマットエンジン状態応答」メッセージを結合部 7 3 0 に送信する（ステップ S 7 6 0 7）。

なお、「フォーマットエンジン状態要求」メッセージを受け取った場合の第 2 の変換部 7 5 1 における処理の流れは、図 7 6 のフローチャートに準ずる。

図 7 7 は、第 3 の変換部 7 7 1 が「フォーマットエンジン状態要求」メッセージを受け取ったときの処理の流れを示すフローチャートである。第 3 の変換部 7 7 1 が「フォーマットエンジン状態要求」メッセージを受け取ると（ステップ S 7 7 0 1）、第 3 の変換部 7 7 1 は、メーラー 7 7 0 が起動しているかどうかを調べる（ステップ S 7 7 0 2）。メーラーが起動している場合、第 3 の変換部 7 7 1 は、メーラー 7 7 0 が「実行中」であることを示す「フォーマットエンジン状態応答」メッセージを作成する（ステップ S 7 7 0 3）。一方、メーラーが起動していない場合

、第3の変換部771は、メーラー770が「停止中」であることを示す「フォーマットエンジン状態応答」メッセージを作成する（ステップS7704）。ステップS7703またはS7704の後、第3の変換部771は、作成した「フォーマットエンジン状態応答」メッセージを結合部730に配送する（ステップS7705）。

以上、図76および図77に示した処理によって、ナビゲータ720に「フォーマットエンジン状態応答」メッセージが返ってくる。図74の説明に戻り、ステップS3の次に、ナビゲータ720は、「フォーマットエンジン状態応答」メッセージを受信する（ステップS4）。続いて、ナビゲータ720は、受信したメッセージに基づいて、ステップS2で選択したフォーマットエンジンが実行中であるか否かを判定する（ステップS5）。

ステップS5において、フォーマットエンジンが実行中でないと判定された場合、ナビゲータ720は、ステップS6の処理をスキップし、ステップS7の処理を行う。一方、フォーマットエンジンが実行中であると判定された場合、ナビゲータ720は、当該フォーマットエンジンに対する「フォーマットエンジン停止」メッセージを結合部730へ送信する（ステップS6）。このメッセージは、当該フォーマットエンジンに対応する変換部へ結合部403によって送信される。

次に、ステップS6の「フォーマットエンジン停止」メッセージを受信した際の各変換部の処理の流れについて説明する。

図78は、第1の変換部741が、「フォーマットエンジン停止」メッセージを受け取った際の処理の流れを表すフローチャートである。第1の変換部741が「フォーマットエンジン停止」メッセージを受け取ると（ステップS7801）、第1の変換部741は、A I TからJavaアプリケーションを1つ選択する（ステップS7802）。続いて、第1の変換部741は、選択したJavaアプリケーションの状態が「D e s t r o y e d」であるか、あるいは読み込む前であるかを調べる（ステップS7803）。選択したJavaアプリケーションの状態が「D e s t r o y e d」であるか、あるいは読み込む前のいずれかでもない場合、第1の変換部741は、選んだJavaアプリケーションの状態を「D e s t r o y e d」に遷移させる（ステップS7804）。選択したJavaアプリケーションの状態が「D e s t r o y e d」であるか、あるいは読み込む前のいずれかである場合、第1の変換部741は、ステップS7804の処理をスキップし、ステップS7805の処理を行う。すなわち、第1の変換部741は、ステップS7802において全てのJavaアプリケーションを選択したか否かを判定する。1つでも選択していないJavaアプリケーションがある場合、第1の変換部741は、ステップS7802の処理を行う。そして、第1の変換部741は、ステップS7802～S7805の処理を繰り返す。一方、全てのJavaアプリケーションを選択した場合、第1の変換部741は、図78の処理を終了する。以上のように、第1の変換部741は、ス

ステップ S 7802～ステップ S 7804までの処理を全てのJavaアプリケーションに対して実施する。

図79は、第2の変換部751が、「フォーマットエンジン停止」メッセージを受け取った際の処理の流れを示すフローチャートである。第2の変換部751が「フォーマットエンジン停止」メッセージを受け取ると（ステップS7901）、第2の変換部751は、A I TからH T M Lデータを1つ選択する（ステップS7902）。続いて、第2の変換部751は、選択したH T M Lデータの状態が「K i l l e d」であるか、あるいは読み込む前であるかを調べる（ステップS7903）。選択したH T M Lデータの状態が「K i l l e d」であるか、あるいは読み込む前のいずれかでもない場合、第2の変換部751は、選んだH T M Lデータの状態を「D e s t r o y e d」に遷移させる（ステップS7904）。選択したH T M Lデータの状態が「K i l l e d」であるか、あるいは読み込む前のいずれかである場合、第2の変換部751は、ステップS7904の処理をスキップし、ステップS7905の処理を行う。すなわち、第2の変換部751は、ステップS7902～ステップS7904までの処理を全てのH T M Lデータに対して実施する（ステップS7905）。

図80は、第3の変換部761が、「フォーマットエンジン停止」メッセージを受け取った際の処理の流れを示すフローチャートである。第3の変換部761が「フォーマットエンジン停止」メッセージを受け取ると（ステップS8001）、第3の変換部761は、メーラー760が起

動しているかどうかを調べる（ステップ S 8002）。メーラー 760 が起動していれば、第 3 の変換部 761 は、メーラー 760 を終了して（ステップ S 8003）、図 80 に示す処理を終了する。一方、メーラー 760 が起動していなければ、第 3 の変換部 761 は、ステップ S 8003 の処理をスキップして図 80 に示す処理を終了する。

以上、図 78～図 80 に示した処理によって、「実行中」の動作状態であったフォーマットエンジンが終了される。図 74 の説明に戻り、ステップ S 5 またはステップ S 6 の次に、ナビゲータ 720 は、ステップ S 2 において、起動要求の対象となっているフォーマットエンジンを除く全てのフォーマットエンジンを選択したか否かを判定する（ステップ S 7）。1 つでも選択していないフォーマットエンジンがある場合、ナビゲータ 720 は、ステップ S 2 の処理を行う。そして、ナビゲータ 720 は、ステップ S 2～S 7 の処理を繰り返す。一方、全てのフォーマットエンジンを選択した場合、ナビゲータ 720 は、起動要求の対象となっているフォーマットエンジンに対する「フォーマットエンジン実行」メッセージを結合部 730 へ送信する（ステップ S 8）。その後、ナビゲータ 720 は、図 74 の処理を終了する。

次に、ステップ S 8 の「フォーマットエンジン実行」メッセージを受信した際の各変換部の処理の流れについて説明する。

図 81 は、第 1 の変換部 741 が、「フォーマットエンジン実行」メッセージを受け取った際の処理の流れを示す

フローチャートである。第1の変換部741が「フォーマットエンジン実行」メッセージを受け取ると（ステップS8101）、第1の変換部741は、実行すべきJavaアプリケーションを決定する（ステップS8102）。この決定は、例えば、AITから最も優先度の高いJavaアプリケーションを選ぶことによって決定される。続いて、第1の変換部741は、ステップS8103において決定したJavaアプリケーションの動作状態が「Active」であるか否かを判定する（ステップS8103）。当該Javaアプリケーションの状態が「Active」でなければ、第1の変換部741は、当該Javaアプリケーションの状態を「Active」に遷移させ（ステップS8104）、図81に示す処理を終了する。一方、当該Javaアプリケーションの状態が「Active」であれば、第1の変換部741は、ステップS8104の処理をスキップし、図81に示す処理を終了する。

なお、「フォーマットエンジン実行」メッセージを受け取った場合の第2の変換部751における処理の流れは、図81のフローチャートに準ずる。

また、「フォーマットエンジン実行」メッセージを受け取った場合の第3の変換部761における処理の流れは、図80のフローチャートに準ずる。

以上のように、図74に示すステップS8において送信される「フォーマットエンジン実行」メッセージによって、起動要求の対象であるフォーマットエンジンが起動される。また、その他のフォーマットエンジン（起動要求の対

象でないフォーマットエンジン) は「実行中」の状態でなくなるので、極小リソースが競合することを回避することができる。

なお、図 74 のステップ S 6 では、ナビゲータ 720 は「フォーマットエンジン停止」メッセージを送信するものとしたが、ナビゲータ 720 は、「フォーマットエンジン一時停止」メッセージを送信してもよい。この場合も、上記その他のフォーマットエンジン(起動要求の対象でないフォーマットエンジン) は「実行中」の状態でなくなるので、極小リソースが競合することを回避することができる。

次に、「フォーマットエンジン一時停止」メッセージを受信した際の各変換部の処理の流れについて説明する。

図 82 は、第 1 の変換部 741 が、「フォーマットエンジン一時停止」メッセージを受け取った際の処理の流れを表すフローチャートである。第 1 の変換部 741 が「フォーマットエンジン一時停止」メッセージを受け取ると(ステップ S 8201)、第 1 の変換部 741 は、A I T から J a v a アプリケーションを 1 つ選択する(ステップ S 8202)。続いて、第 1 の変換部 741 は、選択した J a v a アプリケーションの動作状態が「A c t i v e」であるか否かを判定する(ステップ S 8203)。選択した J a v a アプリケーションの状態が「A c t i v e」である場合、第 1 の変換部 741 は、選んだ J a v a アプリケーションの状態を「P a u s e d」に遷移させる(ステップ S 8204)。選択した J a v a アプリケーションの状態

が「Active」でない場合、第1の変換部741は、ステップS8204の処理をスキップし、ステップS8205の処理を行う。すなわち、第1の変換部741は、ステップS8202～ステップS8204までの処理を全てのJavaアプリケーションに対して実施する（ステップS8205）。

なお、「フォーマットエンジン一時停止」メッセージを受け取った場合の第2の変換部751における処理の流れは、図82のフローチャートに準ずる。

また、「フォーマットエンジン一時停止」メッセージを受け取った場合の第3の変換部761における処理の流れは、「フォーマットエンジン停止」メッセージを受け取った場合と同様である。

（第2の動作例の詳細な動作）

次に、第2の動作例の詳細な動作を説明する。図83は、第2の動作例の動作を行う場合におけるデジタルテレビにおける処理の流れを示すフローチャートである。図83において、まず、ナビゲータ720は、フォーマットエンジンの起動要求を受け付ける（ステップS11）。ステップS11の処理は、第1の動作例におけるステップS1と同様である。

次に、ナビゲータ720は、起動要求の対象であるフォーマットエンジンに対する「フォーマットエンジン実行」メッセージを結合部733へ送信する（ステップS12）。結合部733を介して当該メッセージを受け取った変換部における処理は、上述した第1の動作例と同様である。

ここで、各フォーマットエンジンは、「フォーマットエンジン実行」メッセージに応じて起動した後、必要に応じてリソースの利用をライプラリ 711 に対して要求する。リソースの利用要求を行うか否かは、各フォーマットエンジン毎に予め定められている。以下では、フォーマットエンジンが極小リソースの利用要求を行ったものとして説明する。なお、フォーマットエンジンが極小リソースの利用要求を行わない場合は、フォーマットエンジンはそのまま動作を続ける。

ステップ S12 の次に、ライプラリ 711 は、起動したフォーマットエンジンから極小リソースの利用要求を受け付ける（ステップ S13）。なお、この段階では、当該フォーマットエンジンに対して当該極小リソースの利用が許可されたわけではない。続いて、ライプラリ 711 は、極小リソースの利用要求が重複しているか否かを判定する（ステップ S14）。ここで、「極小リソースの利用要求が重複する」とは、単一の極小リソースに対して複数の利用要求があったことを意味する。従って、異なる 2 つの極小リソースの各々に対して利用要求があつても、極小リソースの利用要求が重複したことにはならない。以下、ステップ S14 における判定処理を詳細に説明する。

ステップ S14において、ライプラリ 711 は、起動要求を行っているフォーマットエンジンをフォーマットエンジン特定部 3905 に問い合わせる。この問い合わせに応じて、フォーマットエンジン特定部 3905 は、プロセス記憶部 3901 を参照することによって、起動要求を行つ

ているフォーマットエンジンを特定する。そして、フォーマットエンジン特定部 3905 は、特定したフォーマットエンジンをライプラリ 711 に通知する。なお、極小リソースをすでに利用しているフォーマットエンジンはすでに特定されている。従って、ライプラリ 711 は、極小リソースをすでに利用しているフォーマットエンジン（これがない場合は重複しないと判定する。）と、フォーマットエンジン特定部 3905 から通知されたフォーマットエンジンとによって、極小リソースの利用要求が重複しているか否かを判定することができる。

ステップ S14において、利用要求が重複していない場合、ライプラリ 711 は、利用要求を行ったフォーマットエンジンに対して極小リソースの利用を許可し（ステップ S19）、図 83 に示す処理を終了する。

一方、ステップ S14において、利用要求が重複している場合、ライプラリ 711 は、優先度情報を取得する（ステップ S15）。具体的には、リソース管理部 733 の優先度情報提供部 3906 に対して、優先度情報の取得を要求する。ここで、優先度情報とは、優先度格納部 3902 に格納されている優先度と、最新起動記憶部 3903 に記憶されている最新起動情報とを含む概念である。つまり、優先度情報提供部 3906 は、ライプラリ 711 からの要求に対して、優先度格納部 3902 に格納されている優先度と、最新起動記憶部 3903 に記憶されている最新起動情報とを送信する。

ライプラリ 711 は、優先度情報提供部 3906 から送

信されてきた情報を用いて、利用を許可するフォーマットエンジンを決定する（ステップS16）。このステップS16によって、図6に示す許可決定部61の機能が実現されることになる。なお、ここでは、ステップS16の処理はライプラリが行うこととしたが、利用を許可するフォーマットエンジンを優先度情報提供部3906が決定し、決定したフォーマットエンジンをライプラリ711へ通知するようにしてもよい。

ステップS16の処理は、例えば以下のように行われる。なお、ここでは、ステップS13において起動要求を行ったフォーマットエンジンをフォーマットエンジンAとし、極小リソースをすでに利用していたフォーマットエンジンをフォーマットエンジンBと呼ぶ。

ライプラリ711は、まず、最新起動情報に基づいて、ユーザが最近起動したフォーマットエンジンがフォーマットエンジンAであるか否かを判定する。そして、ユーザが最近起動したフォーマットエンジンがフォーマットエンジンAであれば、ライプラリ711は、極小リソースの利用を許可するフォーマットエンジンをフォーマットエンジンAに決定する。一方、ユーザが最近起動したフォーマットエンジンがフォーマットエンジンAでなければ、ライプラリ711は、ユーザが最近起動したフォーマットエンジンがフォーマットエンジンBであるか否かを判定する。そして、ユーザが最近起動したフォーマットエンジンがフォーマットエンジンBであれば、ライプラリ711は、極小リソースの利用を許可するフォーマットエンジンをフォーマ

ットエンジンBに決定する。すなわち、ライブラリ711は、極小リソースの利用を許可するフォーマットエンジンを、ユーザが最近起動したフォーマットエンジンに決定する。

さらに、ユーザが最近起動したフォーマットエンジンがフォーマットエンジンAでもフォーマットエンジンBでもない場合、ライブラリ711は、優先度情報記憶部3602に記憶されている情報を用いて極小リソースの利用を許可するフォーマットエンジンを決定する。すなわち、ライブラリ711は、極小リソースの利用を許可するフォーマットエンジンを、優先度情報記憶部3602に記憶されている優先度が高い方のフォーマットエンジンに決定する。

ステップS16の次に、ライブラリ711は、ステップS16において決定したフォーマットエンジンに対して極小リソースの利用を許可する（ステップS17）。続いて、ライブラリ711は、ステップS16において決定したフォーマットエンジンでない方のフォーマットエンジンに対してリソースが剥奪された旨の通知を行う（ステップS18）。具体的には、ライブラリ711は、リソース剥奪通知を当該フォーマットエンジンに送信するように、リソース管理部733のリソース剥奪通知部3907に要求する。これに応じてリソース剥奪通知部3907がリソース剥奪通知を当該フォーマットエンジンに送信する。ステップS18の後、ライブラリ711は、図83に示す処理を終了する。

以上の処理によって、極小リソースの利用が許可されな

かったフォーマットエンジンに対応する変換部にリソース剥奪通知が送られる。変換部は、このリソース剥奪通知に応じて、以下の処理を行う。

図84は、リソース剥奪通知を受け取った変換部における処理の流れを示すフローチャートである。なお、図84に示す処理は、変換部が行うこととするが、フォーマットエンジンが同じ動作を行ってもよい。まず、変換部は、剥奪されたリソースのリソースIDをリソース剥奪通知部3907から受け取る（ステップS8401）。次に、変換部は、フォーマットエンジンが継続して実行可能か否かを判定する（ステップS8402）。ここで、「フォーマットエンジンが継続して実行可能」とは、「リソース（剥奪されたリソース）がなくても実行可能」という意味である。ステップS8402において、フォーマットエンジンが継続実行可能な場合、変換部は、そのまま図84に示す処理を終了する。このとき、リソースが剥奪されたフォーマットエンジンは実行されたままである。

一方、ステップS8402において、フォーマットエンジンが継続実行不可能な場合、変換部は、一時停止状態にすれば実行可能か否かを判定する（ステップS8403）。ステップS8403において、フォーマットエンジンが継続実行不可能であれば、変換部は、フォーマットエンジンを停止する（ステップS8404）。一方、フォーマットエンジンが継続実行可能であれば、変換部は、フォーマットエンジンを一時停止する（ステップS8405）。

なお、図84では、停止したり一時停止したりする対象

をフォーマットエンジンとしている。ここで、フォーマットエンジンが実行するアプリケーションや表示するデータ（上記の Java アプリケーションや HTML データ）に状態が規定されている場合、フォーマットエンジンの動作状態を変更するだけでなく、これらのアプリケーションやデータの状態を変更するようにしてもよい。なお、図 84 で記載した状態は、共通状態である。変換部は、共通状態の変更をフォーマットの動作状態に応じて適切に変換し、フォーマットエンジンの状態を遷移させる。

なお、図 83 では、ライブラリ 711 は、優先度情報として、優先度格納部 3902 に格納されている優先度と、最新起動記憶部 3903 に記憶されている最新起動情報とを用いてステップ S16 における決定を行った。ここで、ライブラリ 711 は、まず最新起動情報のみを取得し、ユーザが最近起動したフォーマットエンジンが上記フォーマットエンジン A でもフォーマットエンジン B でもないとわかった後で、優先度格納部 3902 に格納されている優先度を取得してもよい。また、ライブラリ 711 は、優先度格納部 3902 に格納されている優先度のみを用いてステップ S16 における決定を行ってもよい。以上で、第 2 の動作例の詳細な動作の説明を終了する。

（その他の動作例）

なお、上記の構成例では、図 74～図 84 に示したメッセージの他にも、種々のメッセージのやり取りが行われる。以下、図 74～図 84 に示したメッセージ以外のメッセージをやり取りする場合の変換部の処理を説明する。

図85は、第1の変換部741がアプリケーションマネージャ4503からJavaアプリケーションの状態変化通知を受けた際の処理の流れを示すフローチャートである。第1の変換部741がアプリケーションマネージャ4503からJavaアプリケーションの状態変化の通知を受けると（ステップS8501）、第1の変換部741は、Javaミドルウェア740の共通状態を決定する（ステップS8502）。さらに、第1の変換部741は、1次記憶部109に保存している変化前のJavaミドルウェア740の共通状態と、ステップS8502で決定したJavaミドルウェア740の共通状態とを比較し、変化前後で共通状態が変化したか否かを判定する（ステップS8503）。判定の結果、変化した場合、第1の変換部741は、「フォーマットエンジン状態変化」メッセージを作成し、結合部730に送信する（ステップS8504）。最後に、第1の変換部741は、状態が変化したJavaアプリケーションに関して、「アプリケーション・データ一覧変化」メッセージを作成し、結合部730に送信する（ステップS8505）。

なお、第2の変換部751がHTMLマネージャ5205からHTMLデータの状態変化通知を受けた際の処理の流れは、図85に示す処理に準ずる。

図86は、第1の変換部741が「アプリケーション・データ一覧要求」メッセージを受け取った際の処理の流れを示すフローチャートである。第1の変換部741が「ア

「プリケーション・データ一覧要求」メッセージを受け取ると（ステップS8601）、第1の変換部741は、A I TからJ a v aアプリケーションの一覧を取得する（ステップS8602）。次に、第1の変換部741は、各J a v aアプリケーションの状態をJ a v aミドルウェア740から取得する（ステップS8603）。第1の変換部741は、取得した各J a v aアプリケーションの状態を、アプリケーション・データの共通状態に変換する（ステップS8604）。最後に、第1の変換部741は、「アプリケーション・データ一覧応答」メッセージを作成し、結合部730に送信する（ステップS8605）。

なお、第2の変換部751が「アプリケーション・データ実行」メッセージを受け取った際の処理の流れは、図86に示す処理に準ずる。

図87は、第1の変換部741が、「アプリケーション・データ実行」メッセージを受け取った際の処理の流れを示すフローチャートである。第1の変換部741が「アプリケーション・データ実行」メッセージを受け取ると（ステップS8701）、第1の変換部741は、「アプリケーション・データ実行」メッセージで指定されたJ a v aアプリケーションの動作状態が「A c t i v e」か否かを判定する（ステップS8702）。判定の結果、当該J a v aアプリケーションの状態が「A c t i v e」でない場合、第1の変換部741は、指定されたJ a v aアプリケーションの動作状態を「A c t i v e」に遷移させる（ステップS8703）。一方、当該J a v aアプリケーショ

ンの動作状態が「Active」である場合、第1の変換部741は、図87に示す処理を終了する。

なお、第2の変換部751が「アプリケーション・データ実行」メッセージを受け取った際の処理の流れは、図87に示す処理に準ずる。

図88は、第1の変換部741が、「アプリケーション・データ停止」メッセージを受け取った際の処理の流れを示すフローチャートである。第1の変換部741が「アプリケーション・データ停止」メッセージを受け取ると（ステップS8801）、第1の変換部741は、「アプリケーション・データ停止」メッセージで指定されたJavaアプリケーションの動作状態が「Destroyed」であるか否かを判定する（ステップS8802）。判定の結果、当該Javaアプリケーションの状態が「Destroyed」でない場合、第1の変換部741は、指定されたJavaアプリケーションの動作状態を「Destroyed」に遷移させる（ステップS8803）。一方、当該Javaアプリケーションの動作状態が「Destroyed」である場合、第1の変換部741は、図88に示す処理を終了する。

なお、第2の変換部751が「アプリケーション・データ停止」メッセージを受け取った際の処理の流れは、図88に示す処理に準ずる。

図89は、第1の変換部741が、「アプリケーション・データ一時停止」メッセージを受け取った際の処理の流れを示すフローチャートである。第1の変換部741が「

「アプリケーション・データー時停止」メッセージを受け取ると（ステップS8901）、第1の変換部741は、「アプリケーション・データー時停止」メッセージで指定されたJavaアプリケーションの状態が「Active」であるか否かを判定する（ステップS8902）。判定の結果、Javaアプリケーションの状態が「Active」である場合、第1の変換部741は、指定されたJavaアプリケーションの状態を「Paused」に遷移させる（ステップS8903）。一方、当該Javaアプリケーションの動作状態が「Active」でない場合、第1の変換部741は、図89に示す処理を終了する。

なお、第2の変換部751が「アプリケーション・データー時停止」メッセージを受け取った際の処理の流れは、図89に示す処理に準ずる。

図90は、第3の変換部761がメーラー760が終了したことの通知をメーラー760から受けた際の処理の流れを示すフローチャートである。第3の変換部761がメーラー760が終了したことの通知をメーラー760から受けると（ステップS9001）、第3の変換部761は、「フォーマットエンジン状態変化」メッセージを作成し、結合部730に送信する（ステップS9002）。

図91は、結合部730が「フォーマットエンジン状態変化」メッセージを受け取った場合の結合部730の状態管理部732の処理の流れを示すフローチャートである。まず、結合部730は、「フォーマットエンジン状態変化」メッセージを受け取る（ステップS9101）。次に、

結合部 730 は、受け取ったメッセージ内の Source ID フィールドを参照することによって、状態が変化したフォーマットエンジンを特定し、特定したフォーマットエンジンに対する「フォーマットエンジン状態要求」メッセージを送信する（ステップ S 9102）。その結果、結合部 730 の状態管理部 732 は、「フォーマットエンジン状態応答」メッセージを受け取り（ステップ S 9103）、フォーマットエンジンの状態を 1 次記憶部 109 に記憶する（ステップ S 9104）。

図 92 は、結合部 730 が「アプリケーション・データ一覧変化」メッセージを受け取った場合の結合部 730 の状態管理部 732 の処理の流れを示すフローチャートである。まず、結合部 730 は、「アプリケーション・データ一覧変化」メッセージを受け取る（ステップ S 9201）。次に、結合部 730 は、受け取ったメッセージ内の Source ID フィールドを参照することによって、状態が変化したアプリケーションを実行またはデータを表示しているフォーマットエンジンを特定し、特定したフォーマットエンジンに対する「アプリケーション・データ一覧要求」メッセージを送る（ステップ S 9202）。その結果、結合部 730 の状態管理部 732 は、「アプリケーション・データ一覧応答」メッセージを受け取り（ステップ S 9203）、アプリケーション・データの状態を 1 次記憶部 109 に記憶する。

なお、以上において、ナビゲータ 720 が状態管理部 732 の機能を有することによって、状態管理部 732 を結

合部 730 から削除してもよい。このような構成にすると、ナビゲータ 720 が全てのフォーマットエンジンの状態を管理することになる。

また、本構成例にいて、各フォーマットエンジンは、送られてきたメッセージの発信元に対して、何ら制限をかけない。従って、フォーマットエンジンは、他のフォーマットエンジンの動作状態や、フォーマットエンジンが実行しているアプリケーションや、表示しているデータの状態を変化させることが可能である。例えば、Java ミドルウェア 740 が、HTML ブラウザ 750 に対して、「フォーマットエンジン実行」メッセージを送信することによって、HTML ブラウザ 750 は停止させることができることは明らかである。

また、本構成例では、ROM110 が保存する内容を 2 次記憶部 108 が保存することで、ROM110 を用いない構成とすることも可能である。また、2 次記憶部 108 は、複数のサブ 2 次記憶部で構成し、個々のサブ 2 次記憶部が異なる情報を保存する構成としてもよい。例えば、あるサブ 2 次記憶部は状態管理部 732 が記憶するフォーマットエンジンの状態を記憶し、別のサブ 2 次記憶部は第 1 の変換部 741 が記憶する Java ミドルウェア 740 の状態を記憶し、さらに別のサブ 2 次記憶部は、第 2 の変換部 751 が記憶する HTML ブラウザ 750 の状態を記憶するようにしてもよい。このように、各記憶部を詳細に分割することが可能である。

以上のように、本実施形態によれば、複数のフォーマッ

トエンジンを結合する際、結合部 730 とフォーマットエンジンとを繋ぐ変換部を用いることによって、個々のフォーマットエンジンの実装を大きく改造する必要がない。そのため、少ない工数で複数のフォーマットエンジンを結合できる。

また、結合部 730 が規定する共通状態を用いることによって、結合するフォーマットエンジンが増えた場合であっても、ナビゲータ 720 は、メッセージの送り先を増やすだけよく、追加されたフォーマットエンジンに容易に対応することができる。

また、共通状態を用いることによって、ナビゲータ 720 および結合部 730 は、複数のフォーマットエンジンの状態を容易に把握できる。従って、複数のフォーマットエンジンの管理を容易に行うことができる。具体的には、複数のフォーマットエンジンを同時に制御することが容易になる。

図 93 は、フォーマットエンジンを実行しても、他のフォーマットエンジンに大きな影響を与えない状況の組み合わせを示す図である。図 93 に示すように、全てのフォーマットエンジンが停止中あるいは一時停止中ということは、極小リソースを使用しているフォーマットエンジンがないことを意味する。従って、図 93 に示す状況にある場合、任意のフォーマットエンジンを起動することができる。例えば図 93 に示す状況においては、Java ミドルウェア 740 および HTML ブラウザ 750 を停止することなく、メール 760 を実行することができる。

なお、結合部 730において共通状態が規定されていない場合、ナビゲータ 720は、個々のフォーマットエンジンが規定する独自の状態の組み合わせを把握する必要があることになる。従って、この場合、ナビゲータ 720は、同時実行が可能な状況を判断することが非常に困難となる。特に、本構成例では Java ミドルウェア 740 および HTML ブラウザ 450 については、実行するアプリケーションおよび表示するデータの状態は規定されているが、フォーマットエンジンとしての状態は規定されていない。従って、上記の場合には、ナビゲータ 720は、Java ミドルウェア 740 が実行する Java アプリケーションおよび HTML ブラウザ 450 が表示するデータが取り得る全ての状態の組み合わせを把握する必要がある。また、本構成例で取り上げた Java ミドルウェア 740、HTML ブラウザ 750、およびメール 760 以外のフォーマットエンジンがさらに追加される場合には、追加するフォーマットエンジンが規定する状態の組み合わせをさらに考慮する必要があるので、ますます、複数のフォーマットエンジンの管理が困難となる。

産業上の利用可能性

以上においてはデジタルテレビを実施の形態の一例として記載したが、本発明は、デジタルテレビに限らずパソコン用コンピュータや携帯電話、情報形態端末等、複数のフォーマットエンジンを動作させる情報処理装置に利用することができる。

請求の範囲

1. 異なるフォーマットで記述されたデータをそれぞれ実行するためのフォーマットエンジンを複数格納している情報処理装置であって、

各フォーマットエンジンの動作状態を全てのフォーマットエンジンに共通の表現によって規定した共通状態を予め定義しておき、各フォーマットエンジンの動作を管理するフォーマットエンジン管理手段と、

各フォーマットエンジンに対応して設けられ、共通状態と、フォーマットエンジンの動作状態を各フォーマットエンジン毎に異なる表現によって規定した動作状態である個別状態との対応を予め定義しておき、任意の個別状態となるようにフォーマットエンジンの動作を制御する動作制御手段とを備え、

前記フォーマットエンジン管理手段は、あるフォーマットエンジンを所定の共通状態に変化させる場合、当該所定の共通状態を示す共通状態情報を含むメッセージを、当該フォーマットエンジンに対応して設けられた動作制御手段へ送信し、

前記動作制御手段は、前記フォーマットエンジン管理手段からメッセージが送信されてきた場合、当該メッセージに含まれる共通状態情報により示される共通状態に対応する個別状態となるように、当該フォーマットエンジンを制御する、情報処理装置。

2. 各フォーマットエンジンに対応して設けられ、フォー

マットエンジンの個別状態と、当該個別状態に対応する共通状態との組によって構成されるテーブル格納するテーブル格納手段をさらに備え、

前記動作制御手段は、前記テーブルを参照することによって共通状態から個別状態を決定する、請求項1に記載の情報処理装置。

3. 各フォーマットエンジンに対応して設けられ、フォーマットエンジンの個別状態を取得し、取得した個別状態に対応する共通状態を示す共通状態情報を前記フォーマットエンジン管理手段へ送信する個別状態取得手段をさらに備え、

前記フォーマットエンジン管理手段は、前記個別状態取得手段から出力された共通状態情報により示される共通状態に基づいて、各フォーマットエンジンの動作を管理する、請求項1に記載の情報処理装置。

4. フォーマットエンジンが実行中に利用するリソースであって、複数のフォーマットエンジンが同時に利用することが不可能なリソースである極小リソースをさらに備え、

前記個別状態取得手段は、フォーマットエンジンから取得した個別状態が極小リソースを利用している動作状態を示す場合、当該フォーマットエンジンの共通状態情報として、所定の状態を示す共通状態情報を前記フォーマットエンジン管理手段へ出力するとともに、フォーマットエンジンから取得した個別状態が極小リソースを利用していない動作状態を示す場合、当該フォーマットエンジンの共通状態情報として、前記所定の状態以外の状態を示す共通状態

情報を前記フォーマットエンジン管理手段へ出力し、

前記フォーマットエンジン管理手段は、共通状態情報が前記所定の状態を示すフォーマットエンジンが1つのみになるように、各フォーマットエンジンの動作を管理する、請求項3に記載の情報処理装置。

5. 前記フォーマットエンジン管理手段は、

フォーマットエンジンを起動するための起動要求を受け付ける起動受付手段と、

前記起動受付手段が起動要求を受け付けたことに応じて、前記状態取得手段から各フォーマットエンジンの共通状態情報を取得する共通状態取得手段と、

前記共通状態取得手段によって取得された共通状態情報が実行中を示すフォーマットエンジンがある場合、当該フォーマットエンジンの動作を停止させるメッセージを当該フォーマットエンジンに対応して設けられた動作制御手段へ送信する動作停止手段と、

前記動作停止手段によってフォーマットエンジンの動作が停止された後、起動要求に対応するフォーマットエンジンを起動させるメッセージを当該フォーマットエンジンに対応して設けられた動作制御手段へ送信する起動手段とを含む、請求項4に記載の情報処理装置。

6. フォーマットエンジンが実行中に利用するリソースであって、複数のフォーマットエンジンが同時に利用することが不可能なリソースである極小リソースと、

フォーマットエンジンの要求に応じてフォーマットエンジンに対してリソースの利用を許可するリソース制御手段

と、

極小リソースを利用する場合における各フォーマットエンジン間の優先度を示す優先度情報を格納する優先度情報格納手段と、

極小リソースを利用する要求が複数のフォーマットエンジン間で重複する場合、前記優先度情報に基づいて、当該極小リソースの利用を許可すべきフォーマットエンジンを決定する許可決定手段とをさらに備え、

前記リソース制御手段は、極小リソースを利用する要求が複数のフォーマットエンジン間で重複する場合、前記許可決定手段によって決定されたフォーマットエンジンのみに当該極小リソースの利用を許可し、極小リソースを利用する要求が複数のフォーマットエンジン間で重複しない場合、当該要求を行ったフォーマットエンジンに当該極小リソースの利用を許可する、請求項1に記載の情報処理装置。

7. 前記極小リソースは複数設けられ、

前記リソース制御手段は前記極小リソースに対応して複数設けられる、請求項6に記載の情報処理装置。

8. 異なるフォーマットで記述されたデータをそれぞれ実行するためのフォーマットエンジンを格納している情報処理装置のコンピュータで実行可能なプログラムであって、

前記コンピュータを、

フォーマットエンジンの動作状態を全てのフォーマットエンジンに共通の表現によって規定した共通状態を予め定義しておき、各フォーマットエンジンの動作を管理する

フォーマットエンジン管理手段と、

各フォーマットエンジンに対応して設けられ、共通状態と、フォーマットエンジンの動作状態を各フォーマットエンジン毎に異なる表現によって規定した動作状態である個別状態との対応を予め定義しておき、任意の個別状態となるようにフォーマットエンジンの動作を制御する動作制御手段として機能させ、

前記フォーマットエンジン管理手段は、あるフォーマットエンジンを所定の共通状態に変化させる場合、当該所定の共通状態を示す共通状態情報を含むメッセージを、当該フォーマットエンジンに対応して設けられた動作制御手段へ送信し、

前記動作制御手段は、前記フォーマットエンジン管理手段からメッセージが送信されてきた場合、当該メッセージに含まれる共通状態情報により示される共通状態に対応する個別状態となるように、当該フォーマットエンジンを制御する、プログラム。

図 1

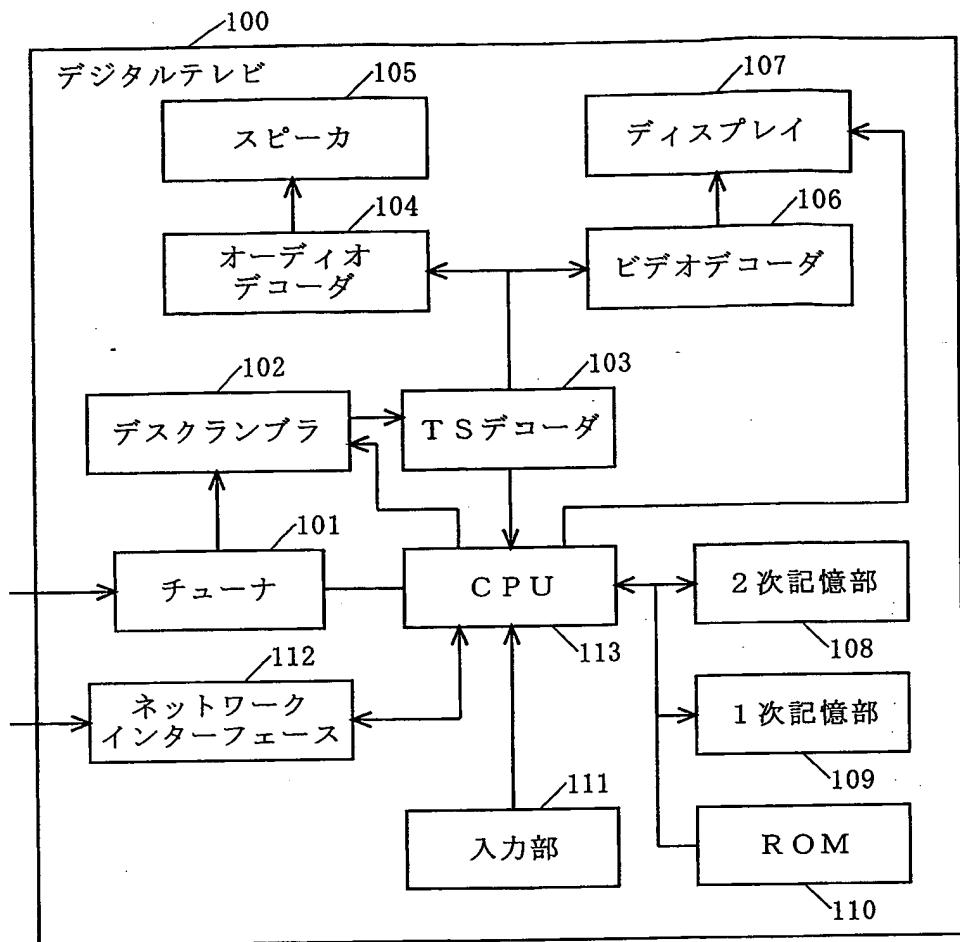


図 2

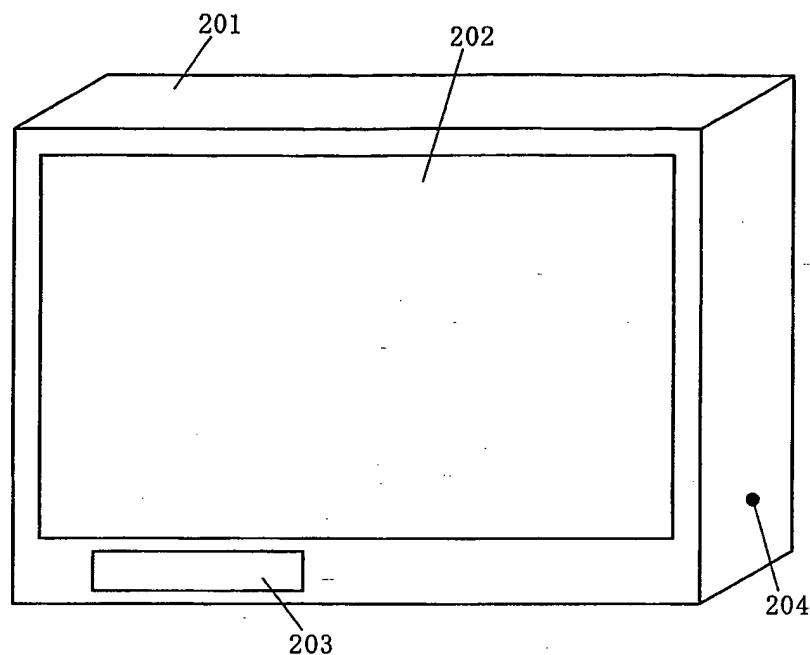


図 3

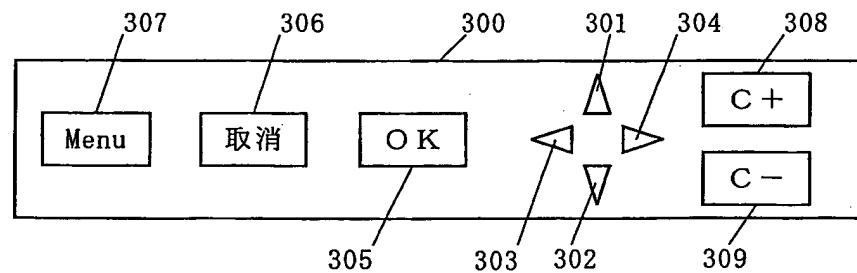


図 4

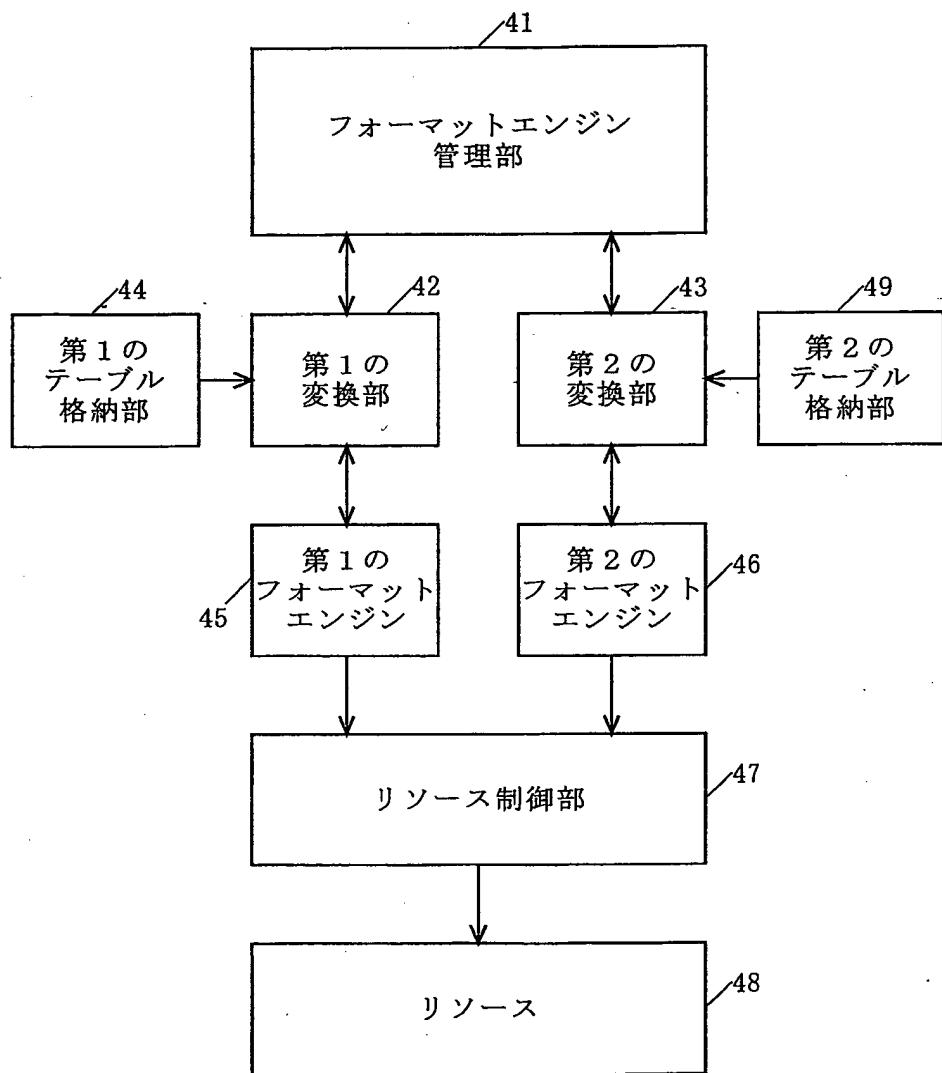


図 5

(a)

共通状態	第 1 のフォーマット エンジンの 個別状態
実行中	動作中
終了	停止中 一時停止中

(b)

共通状態	第 2 のフォーマット エンジンの 個別状態
実行中	起動中
終了	起動中でない

図 6

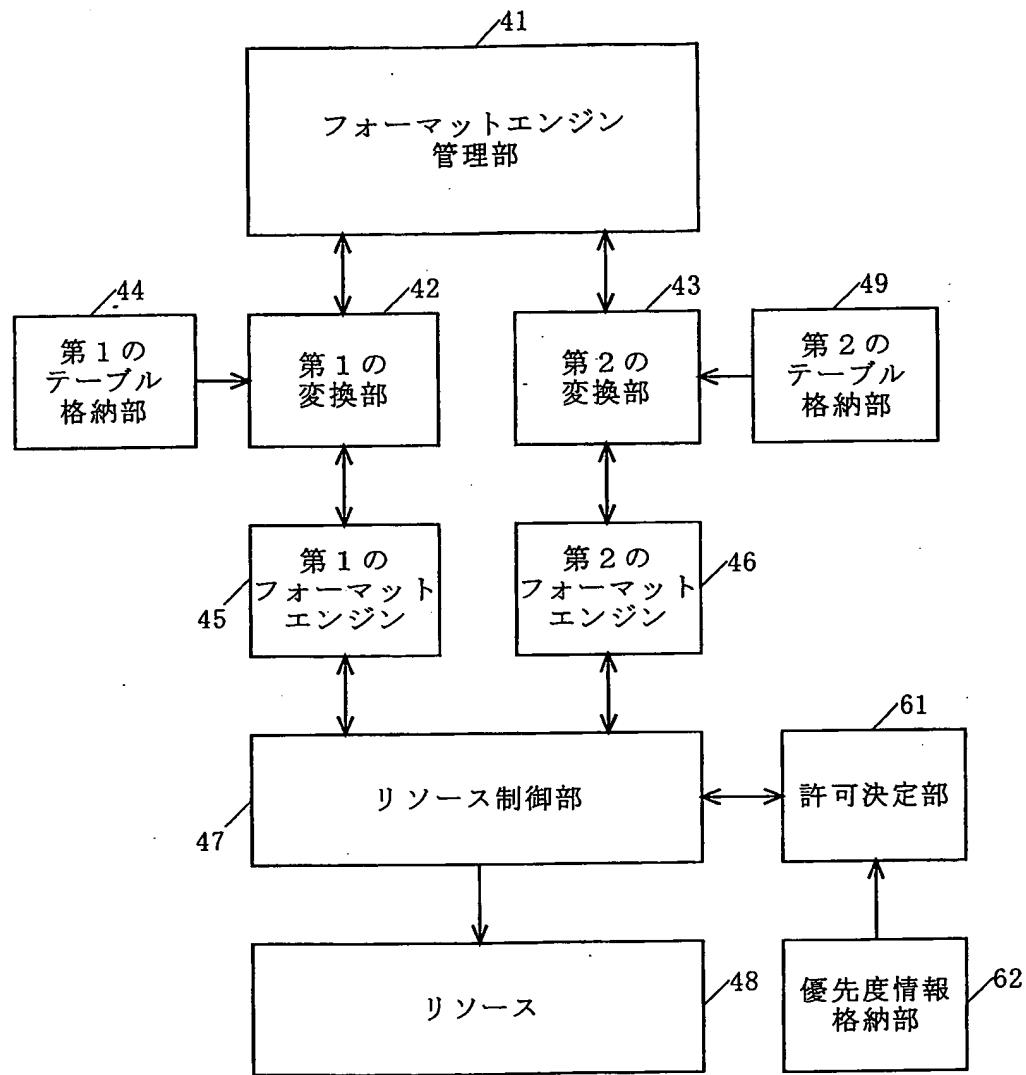


図 7

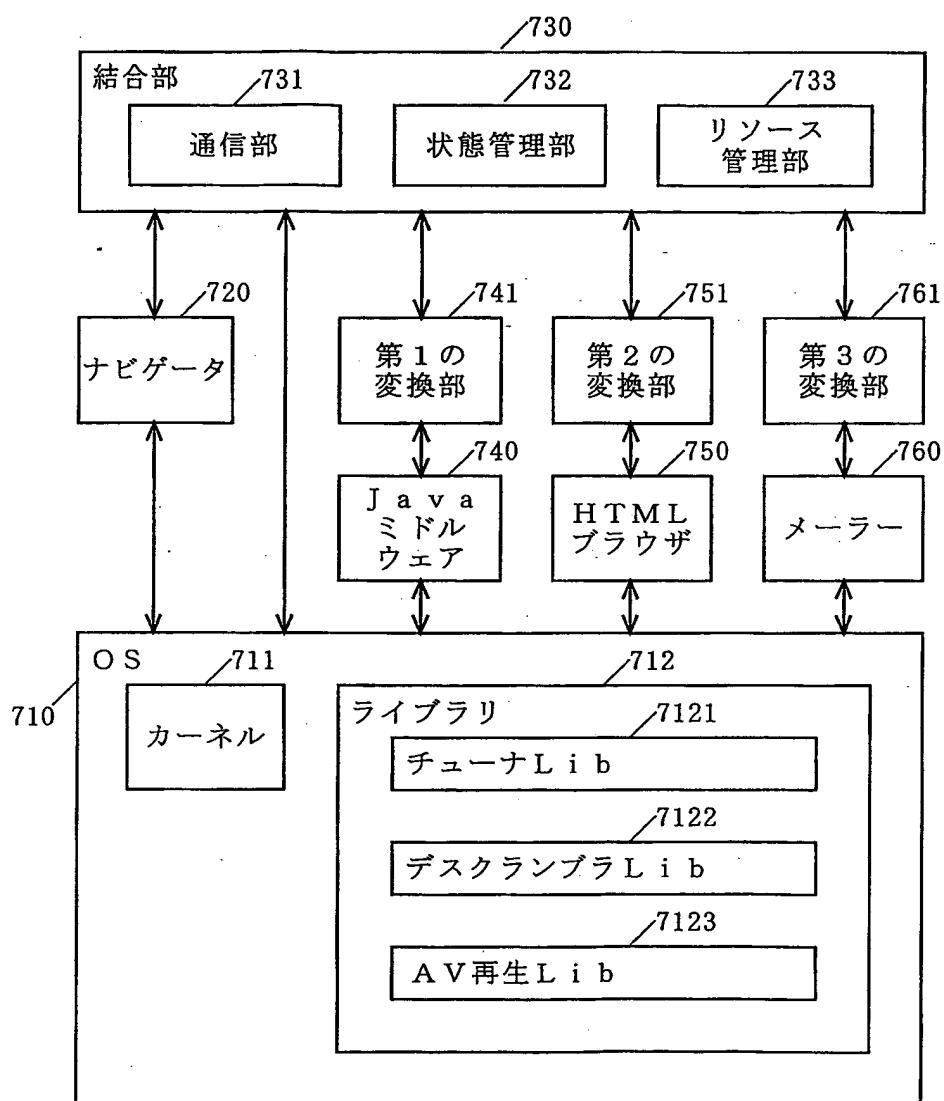


図 8

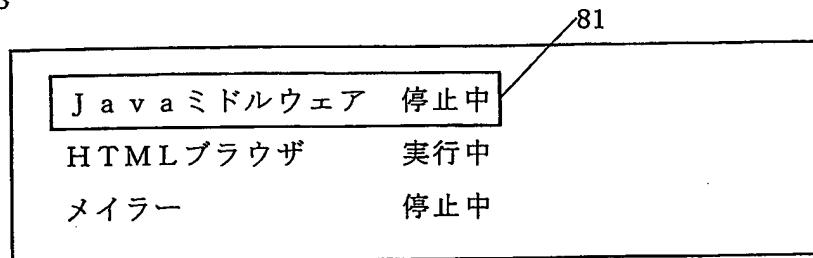


図 9

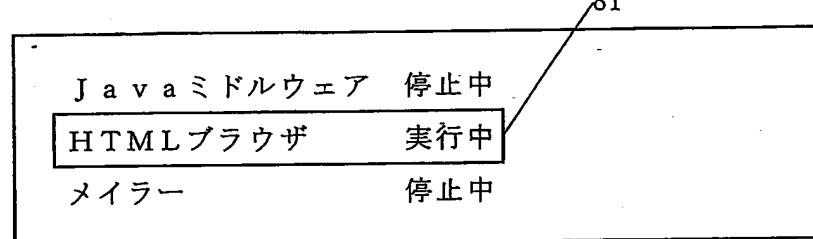


図 10

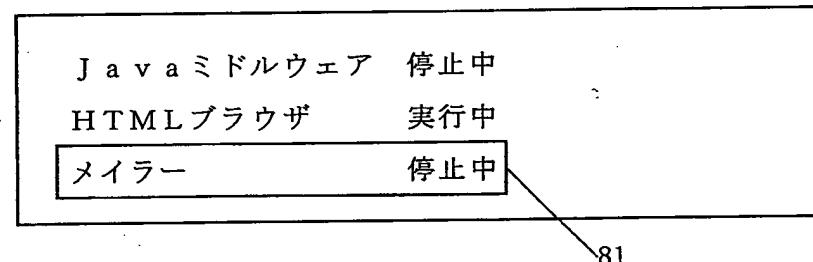


図 1 1

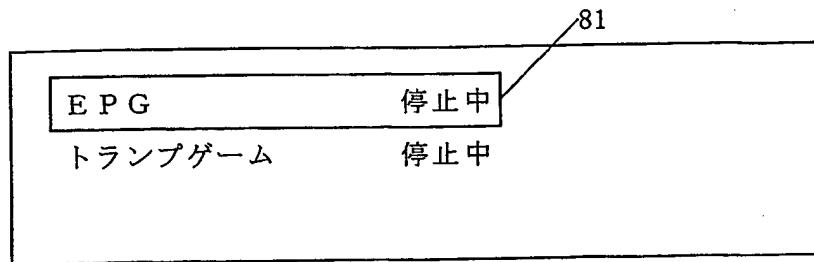


図 1 2

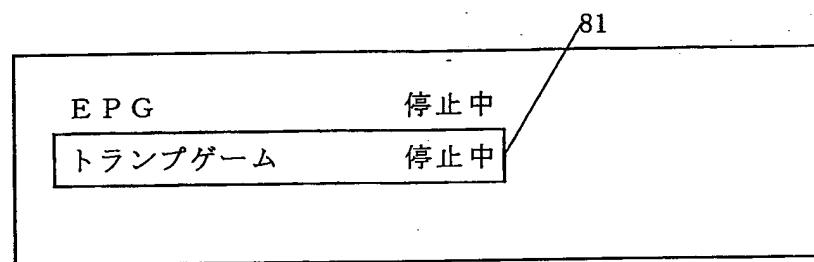


図 1 3

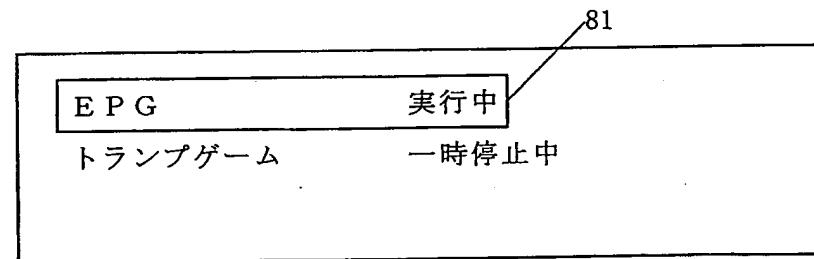


図 1 4

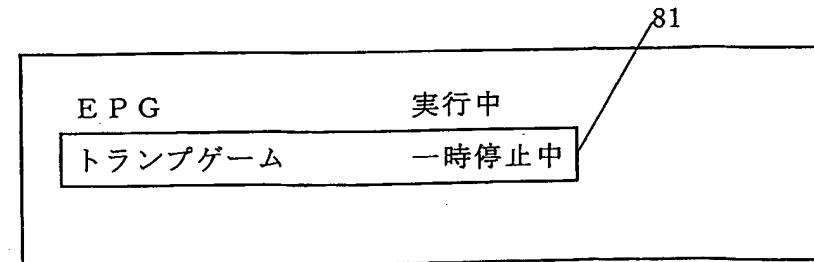


図 1 5

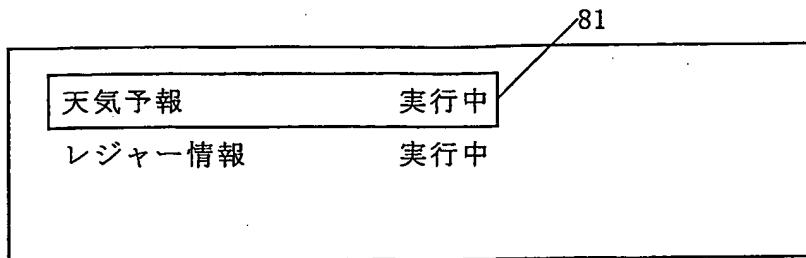
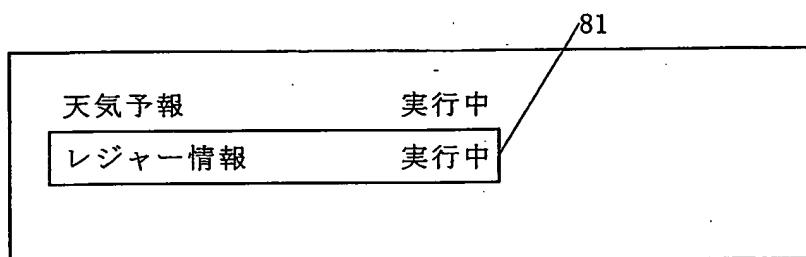


図 1 6



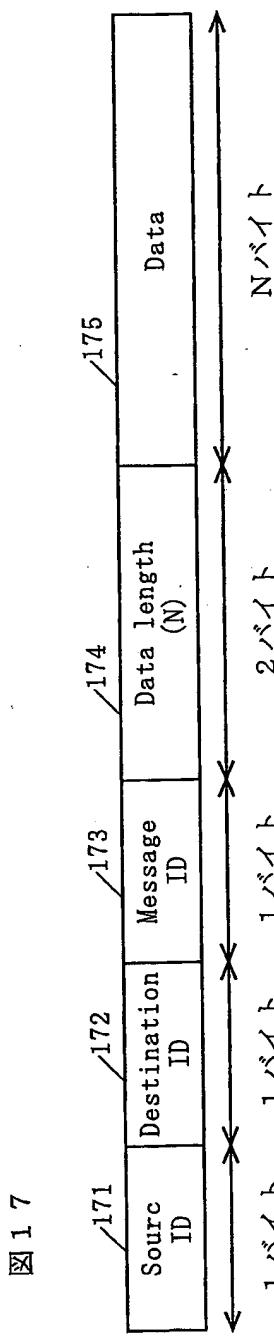


図 18

サブプログラム	S I D
ALL	0
ナビゲータ	1
Java ミドルウェア	2
HTML ブラウザ	3
メイラー	4
結合部	5

図 19

メッセージ	M I D	フォーマット I D
フォーマットエンジン状態要求	1	なし
フォーマットエンジン状態応答	2	F 0 1
フォーマットエンジン状態変化	3	なし
フォーマットエンジン実行	4	なし
フォーマットエンジン終了	5	なし
フォーマットエンジン一時停止	6	なし
アプリケーション・データ一覧要求	1 1	なし
アプリケーション・データ一覧応答	1 2	F 0 2
アプリケーション・データ一覧変化	1 3	なし
アプリケーション・データ実行	1 4	F 0 3
アプリケーション・データ終了	1 5	F 0 3
アプリケーション・データ一時停止	1 6	F 0 3
プライベート	2 1	—

図 2 0

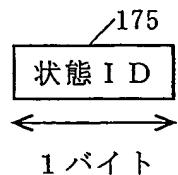


図 2 1

状態名	意味	動作状態 ID
実行中	極小リソースを使用中	1
一時停止中	共有可能リソースを使用中	2
停止中	リソース未使用	3

図 2 2

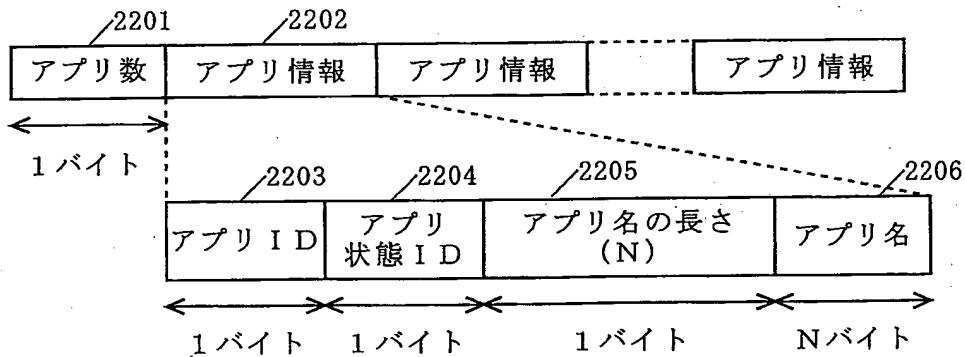


図 2 3

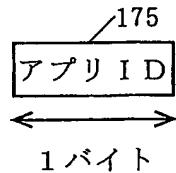


図 2 4

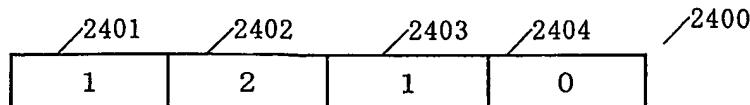


図 2 5

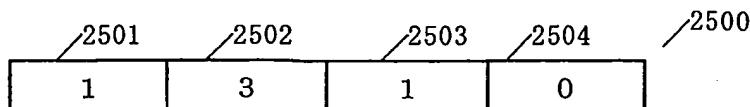


図 2 6

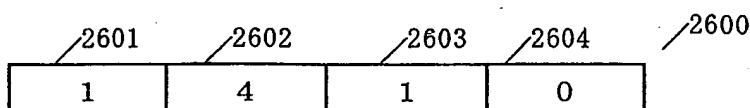


図 2 7

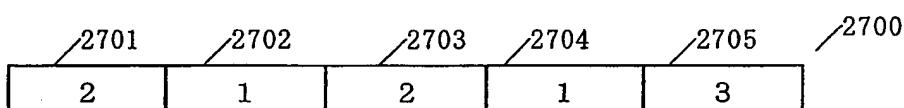


図 2 8

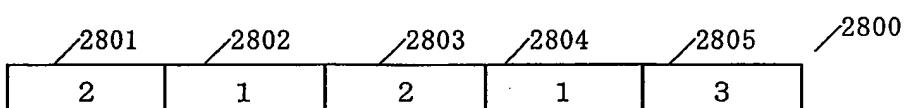


図 2 9

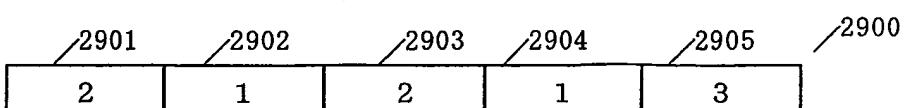


図 3 0

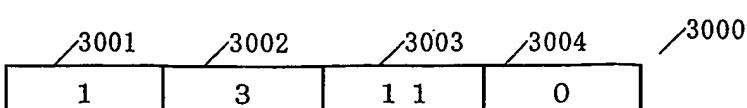


図 3 1

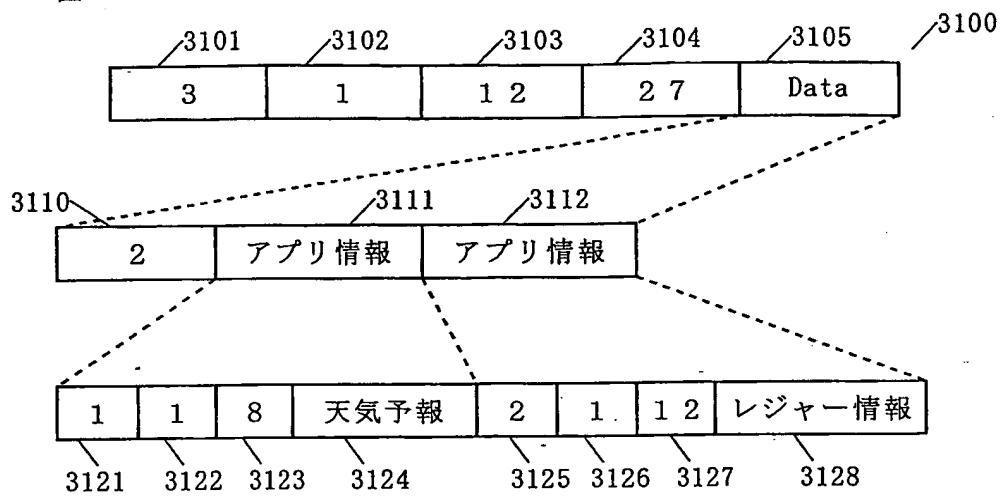


図 3 2

F I D	フォーマットエンジン名	状態 I D	アプリ情報
2	J a v a ミドルウェア	3	0x5678
3	H T M L ブラウザ	3	0x7162
4	メイラー	3	NULL

図 3 3

A I D	アプリ名	状態 I D
1	E P G	3
2	トランプゲーム	3

図 3 4

データ I D	アプリ名	状態 I D
1	天気予報	1
2	レジャー情報	1

図 3 5

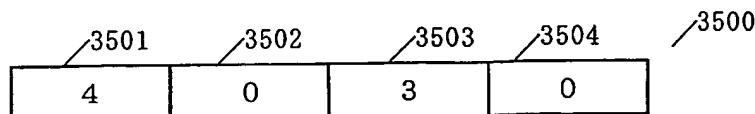


図 3 6

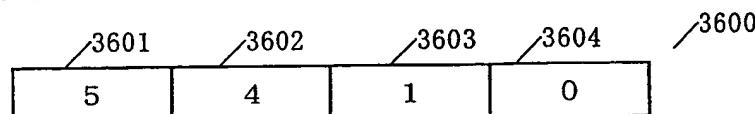


図 3 7

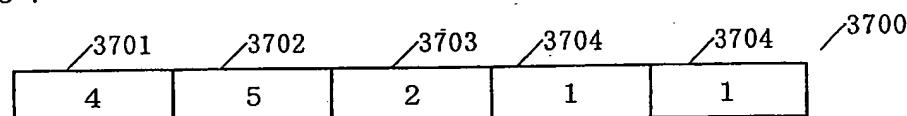


図 3 8

F I D	フォーマットエンジン名	状態 I D	アプリ情報
2	J a v a ミドルウェア	3	0x5678
3	H T M L ブラウザ	3	0x7162
4	メイラー	1	NULL

図 3 9

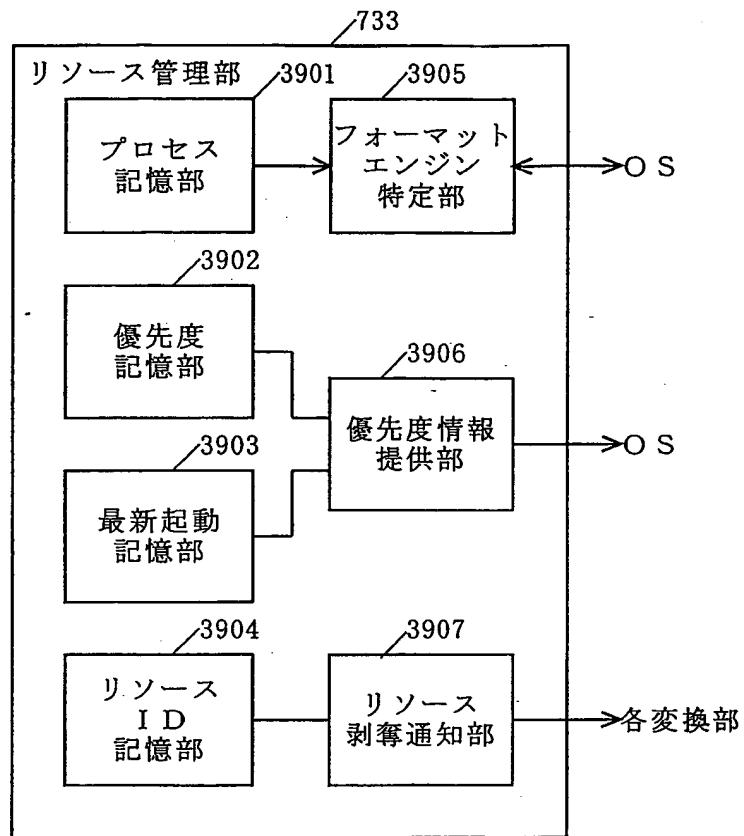


図 4 0

F I D	プロセス I D
2	1 0 0
3	1 1 0
4	1 2 0

図 4 1

F I D	優先度
2	2
3	1
4	3

図 4 2

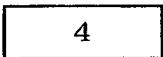


図 4 3

R I D	リソース
1	チューナ
2	TSデコーダ
3	オーディオデコーダ
4	ビデオデコーダ
5	ネットワークインターフェース

図 4 4

```
#define RESOURCE_ID_TUNER 1
#define RESOURCE_ID_TSDECODER 2
#define RESOURCE_ID_AUDIODECODER 3
#define RESOURCE_ID_VIDEODECODER 4
#define RESOURCE_ID_NETWORK 5
```

図 4 5

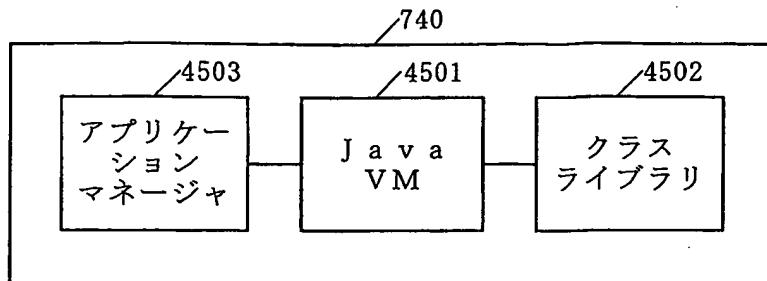


図 4 6

A I D	制御情報	アプリ名	アプリ優先度
1	autostart	E P G	6 4
2	present	トランプゲーム	3 2

図 4 7

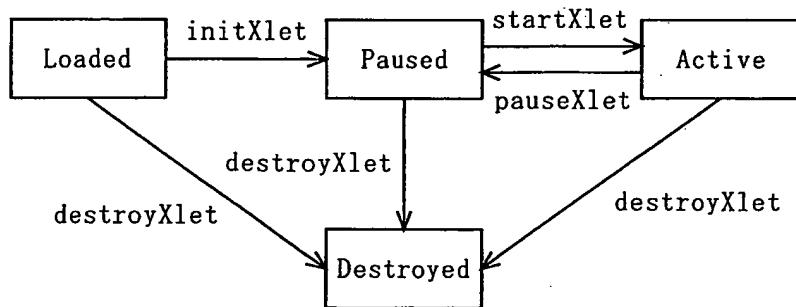


図 4 8

A I D	制御情報	アプリ名	アプリ優先度
1	kill	E P G	6 4
2	present	トランプゲーム	3 2

図 4 9

アプリケーションの状態	Java ミドルウェアの状態
少なくとも 1 つの アプリケーションの状態が 「Active」	実行中
「Active」状態の アプリケーションがなく、かつ、 少なくとも 1 つの アプリケーションの状態が 「Paused」	一時停止中
上記以外	停止中

図 5 0

アプリケーションの状態	結合部が規定する状態
Active	実行中
Paused	一時停止中
上記以外	停止中

図 5 1

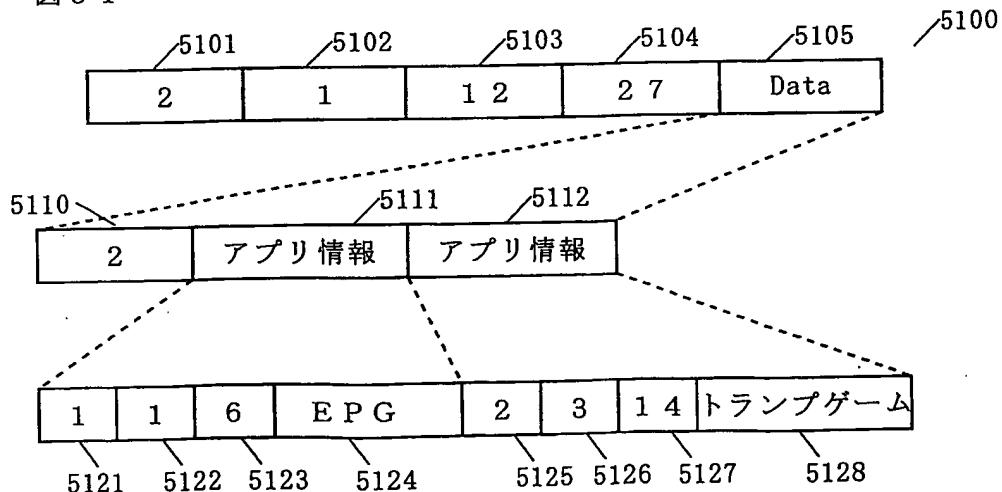


図 5 2

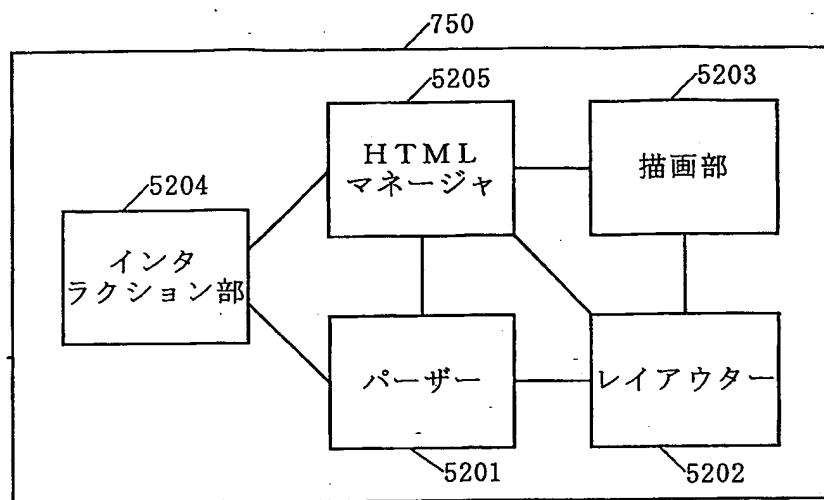


図 5 3

```
<?xml version="1.0"?>
<!DOCTYPE html PUBLIC "-//DVB//DTD XHTML DVB-
HTML 1.0//EN"
"http://www.dvb.org/mhp/dtd/dvbhtml-1-0.dtd">
<html xmlns="http://www.w3c.org/1999/xhtml">
<head>
<link rel="stylesheet" href="sampleCSS.css" type="text/css">
</head>
<body id="body">
<H1>各国のWeather forecast サービス紹介</H1> 5311
<a class="param1" href="dvb://1.2.1/Japan_Weather.html">
Japan</a> 5312
<br></br>
<a class="param2" href="dvb://1.2.2/USA_Weather.html">
USA</a> 5313
<br></br>
<a class="param3" href="dvb://1.2.3/Germany_Weather.html">
Germany</a> 5314
</body>
</html>
```

図 5 4

```
H1 {position: absolute; top: 50px; left: 50px; }
a.param1 {position: absolute; top: 200px; left: 100px; }
a.param2 {position: absolute; top: 300px; left: 100px; }
a.param3 {position: absolute; top: 400px; left: 100px; }
```

図 5 5

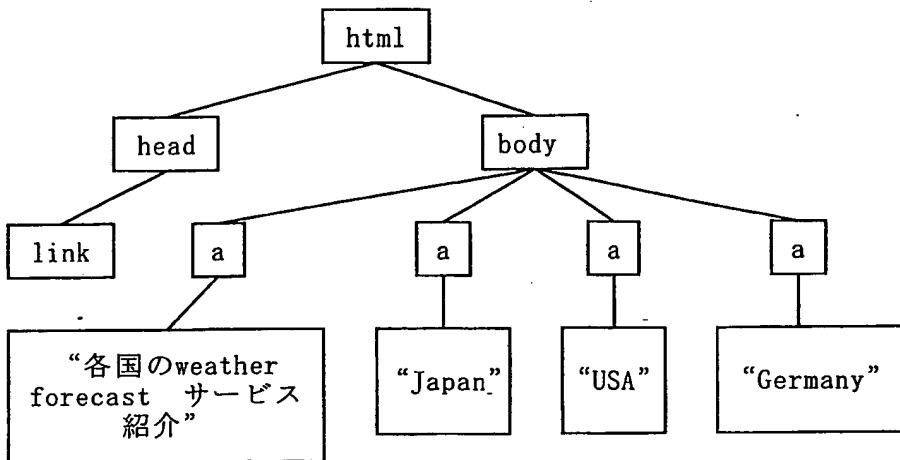


図 5 6

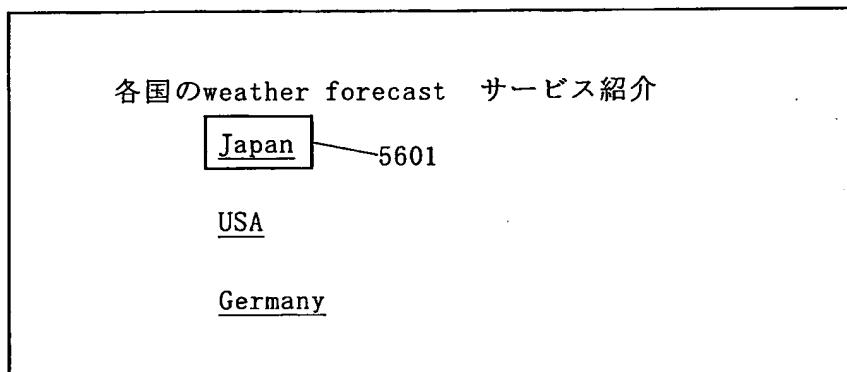


図 5 7

天気予報

東京 くもり

大阪 はれ

福岡 雨

図 5 8

```
<?xml version="1.0"?>
<!DOCTYPE html PUBLIC "-//DVB//DTD XHTML DVB-HTML
1.0//EN"
"http://www.dvb.org/mhp/dtd/dvbhtml-1-0.dtd">
<html xmlns="http://www.w3c.org/1999/xhtml">
<head>
<script type="text/javascript"> 5801
<!--
function ChangeDOMTree(){

}
//--
</script>
</head>
<body>
<table>
<tr>
<td>海 </td>
</tr>
<tr>
<td>山</td>
</tr>
</table> 5802
<form>
<input type="button" value="詳細情報"
onclick= ChangeDOMTree() >
</input>
</form>
</body>
<html>
```

図 5 9

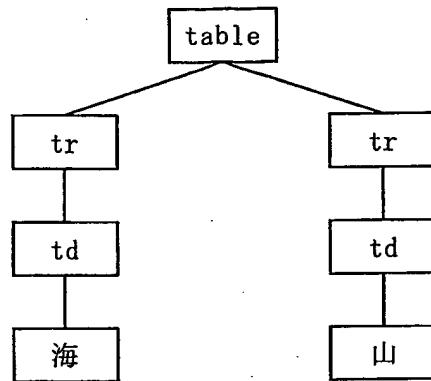


図 6 0

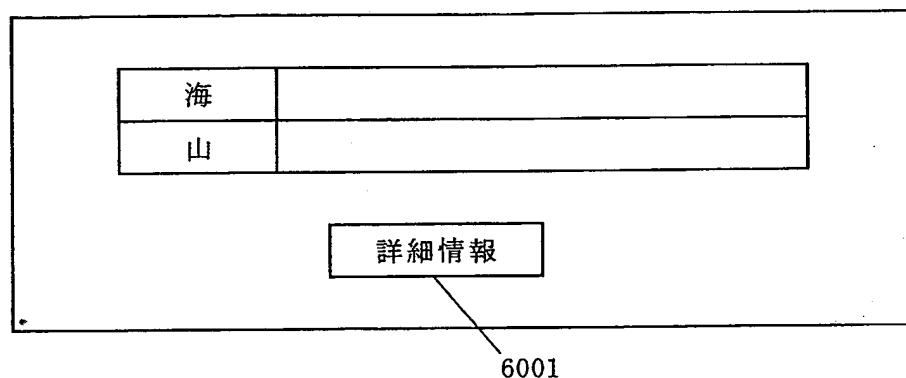


図 6 1

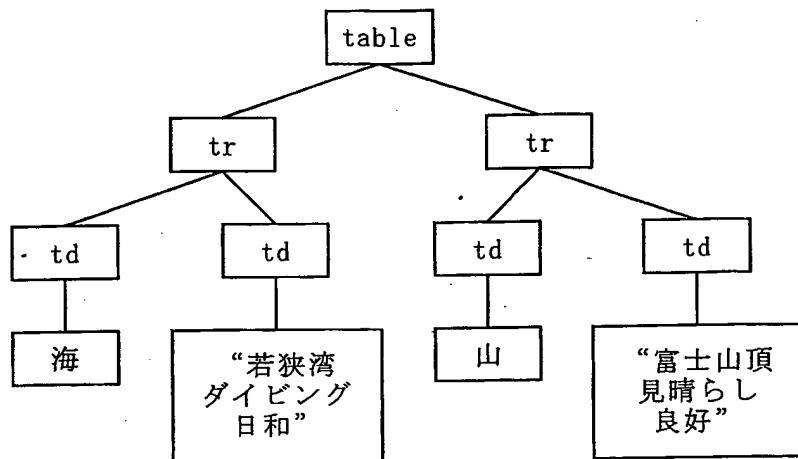


図 6 2

海	若狭湾ダイビング日和
山	富士山頂見晴らし良好

詳細情報

図 6 3

データ I D	制御情報	データ名	アプリ優先度
1	autostart	天気予報	5 4
2	present	レジャー情報	2 2

図 6 4

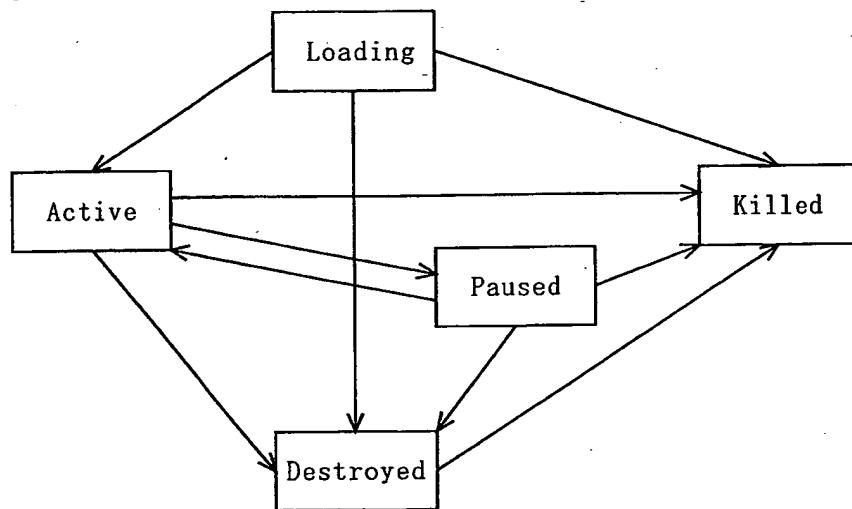


図 6 5

データ I D	制御情報	データ名	アプリ優先度
1	kill	天気予報	5 4
2	present	レジャー情報	2 2

図 6 6

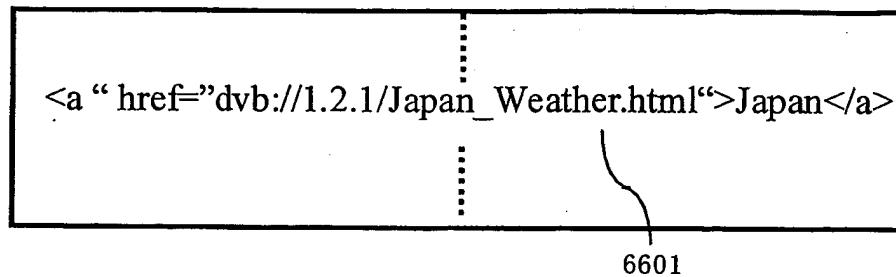


図 6 7

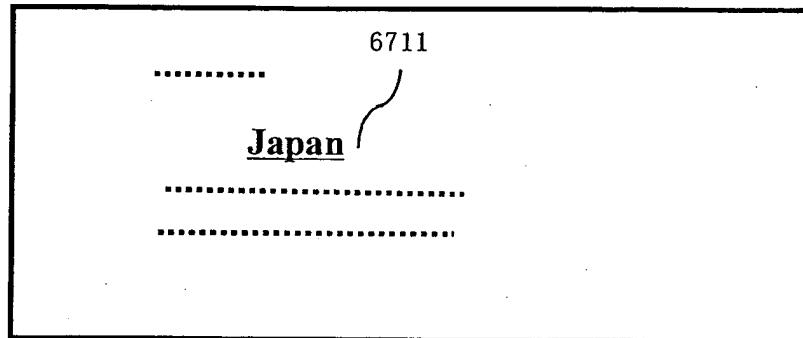


図 6 8

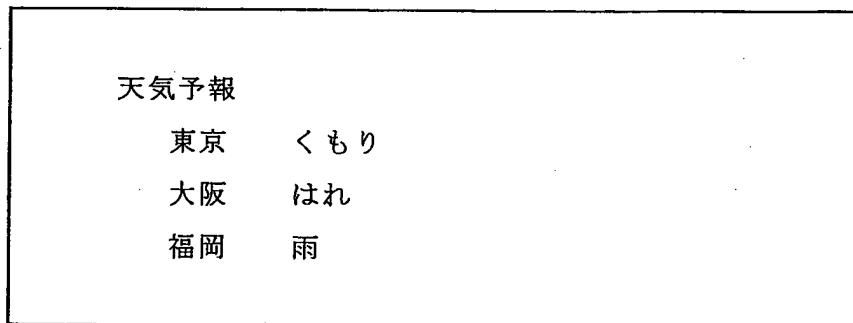


図 6 9

HTMLデータの状態	HTMLブラウザの状態
少なくとも1つのHTMLデータの状態が「Active」	実行中
「Active」状態のHTMLデータがなく、かつ、少なくとも1つのHTMLデータの状態が「Paused」	一時停止中
上記以外	停止中

図 7 0

HTMLデータの状態	結合部が規定する状態
Active	実行中
Paused	一時停止中
上記以外	停止中

図 7 1

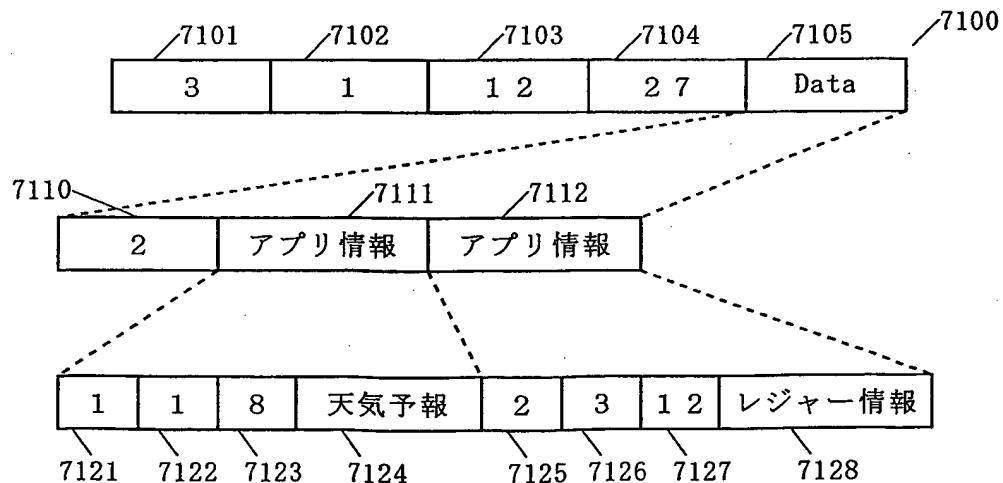


図 7 2

マイラーの状態	結合部が規定する状態
起動中	実行中
停止中	停止中

図 7 3

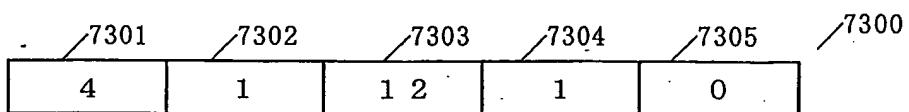


図 7 4

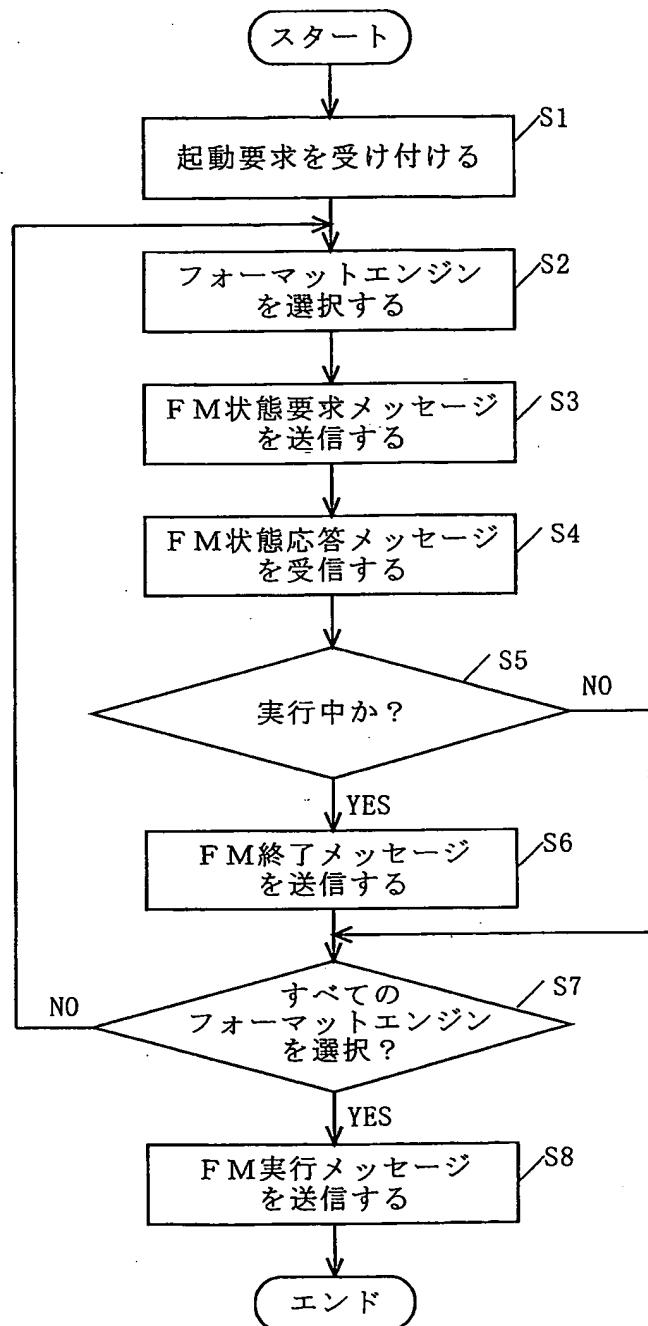


図 7 5

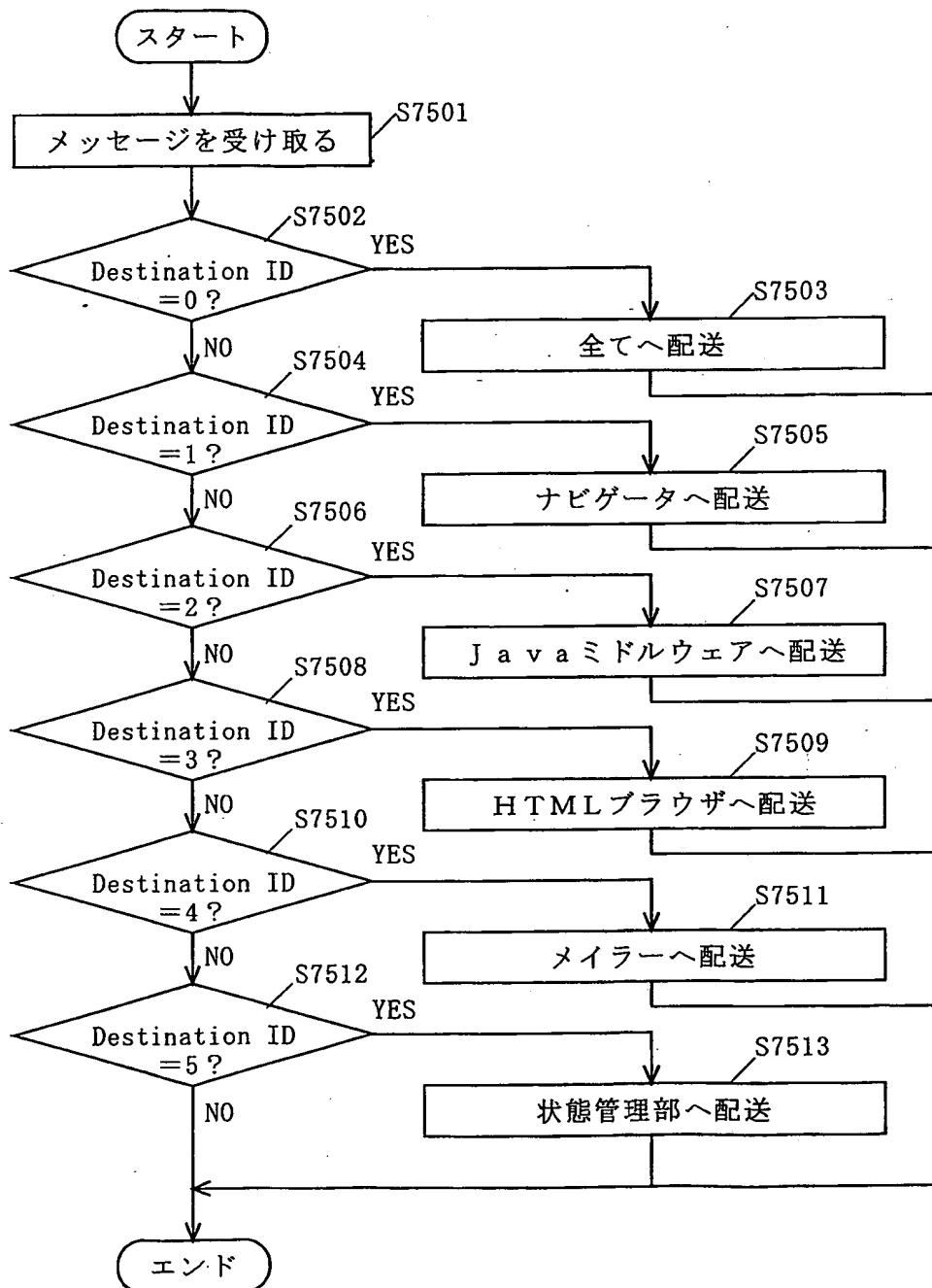


図 7 6

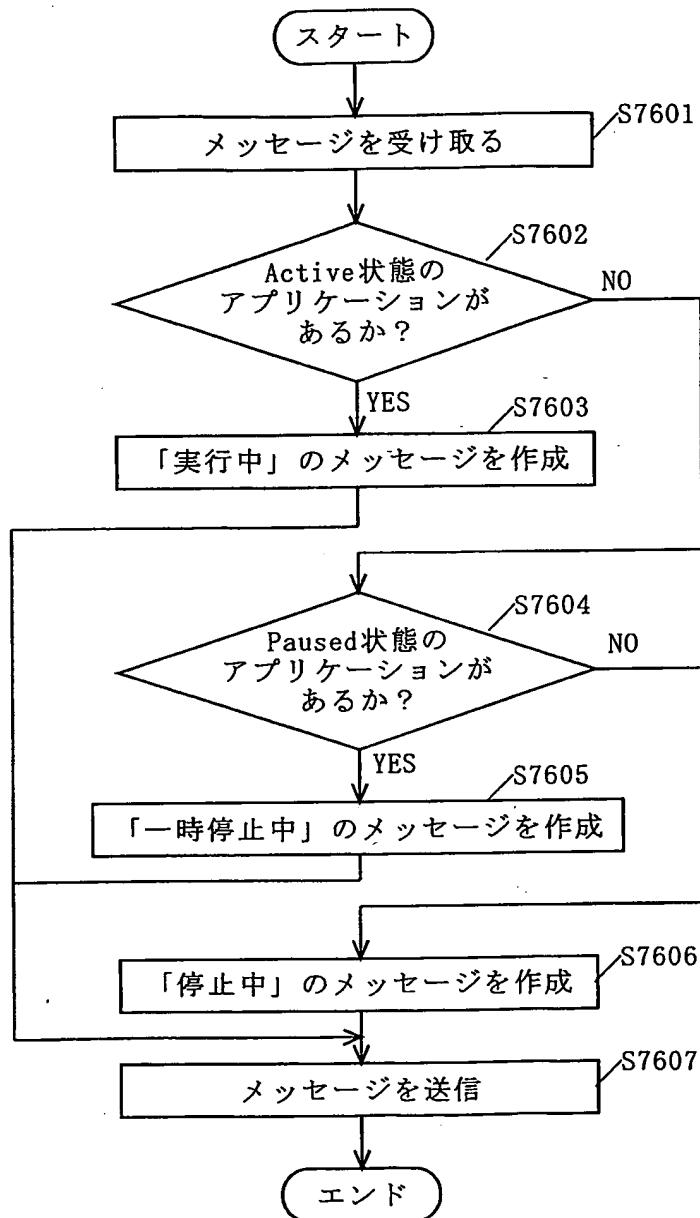


図 7 7

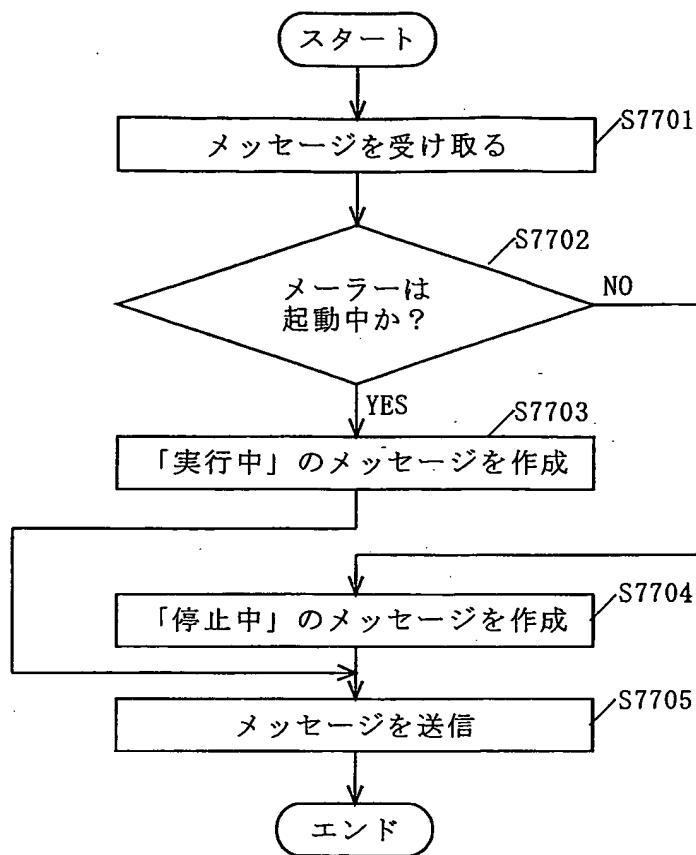


図 7 8

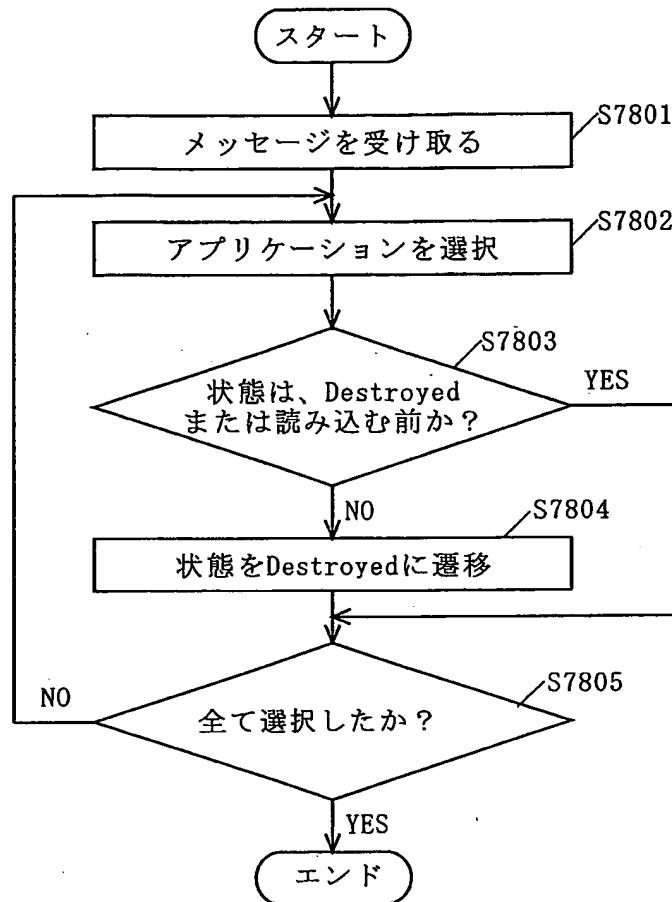


図 7 9

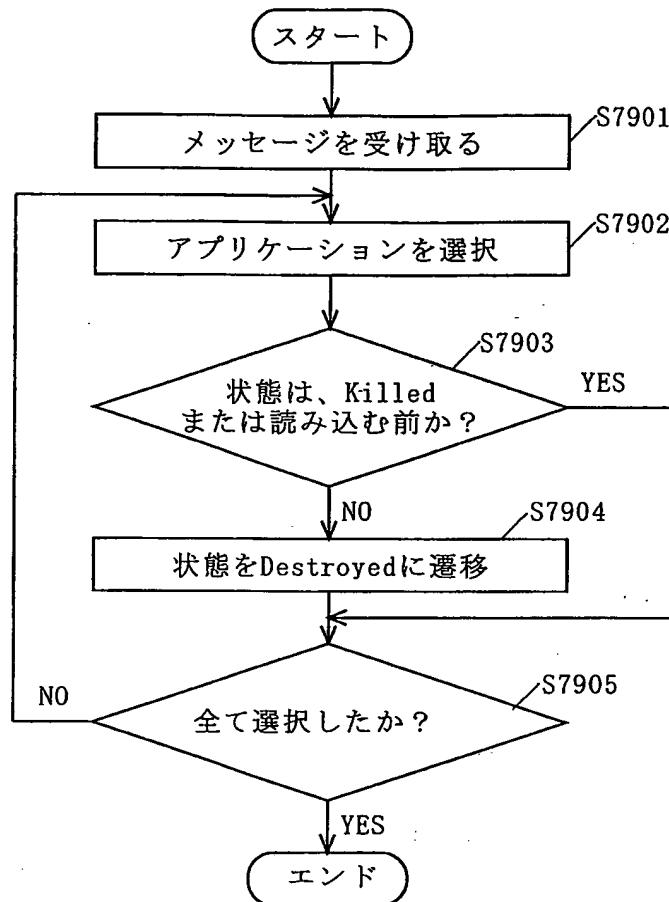


図 8 0

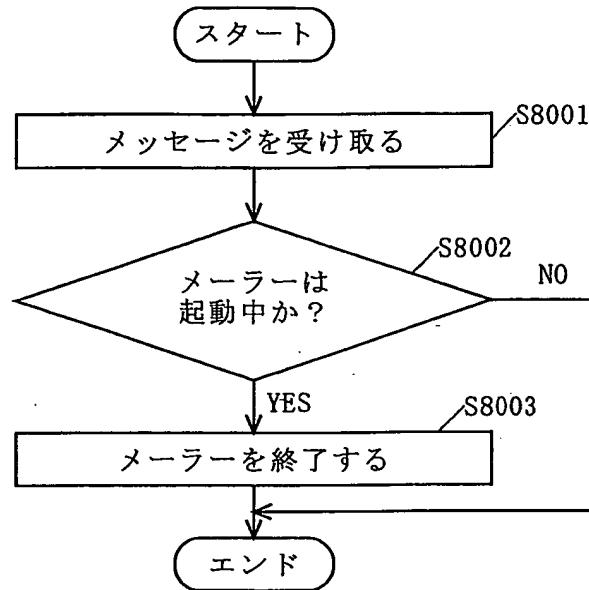


図 8 1

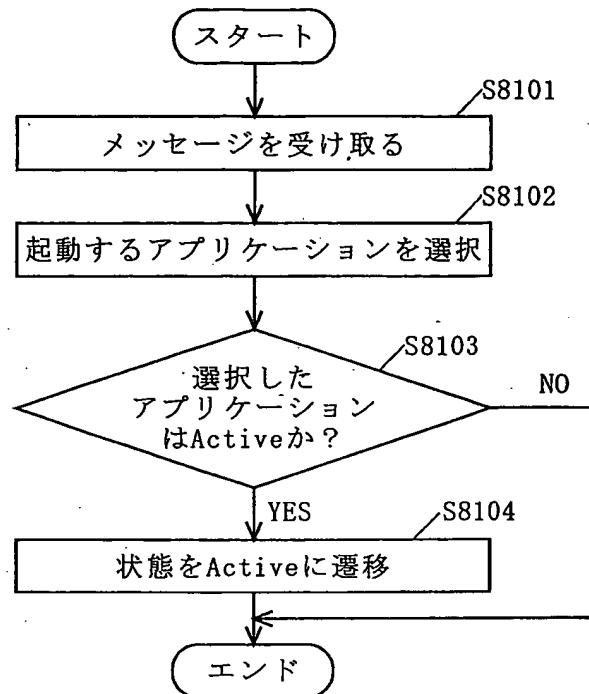


図 8 2

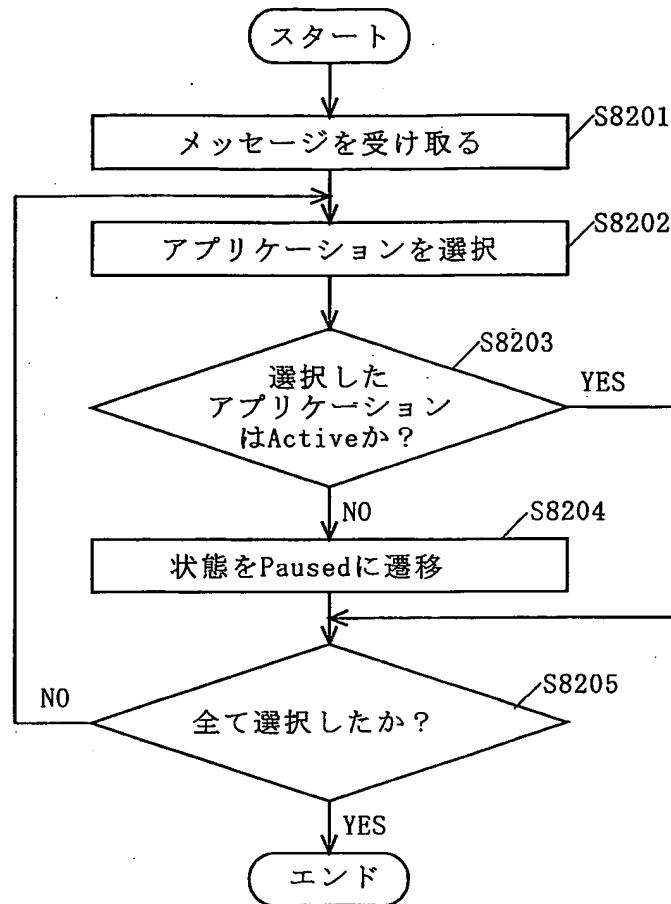


図 8 3

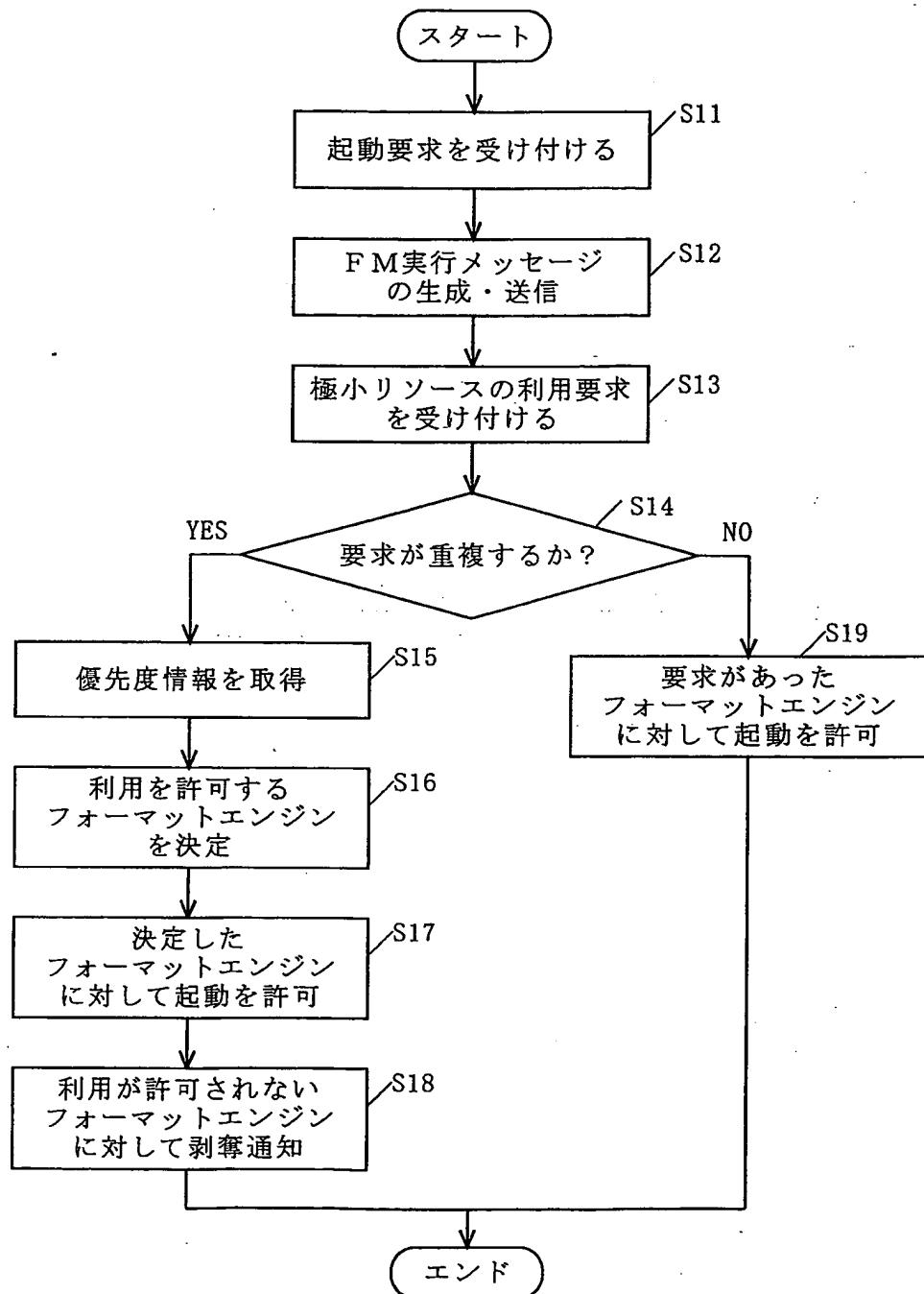


図 8 4

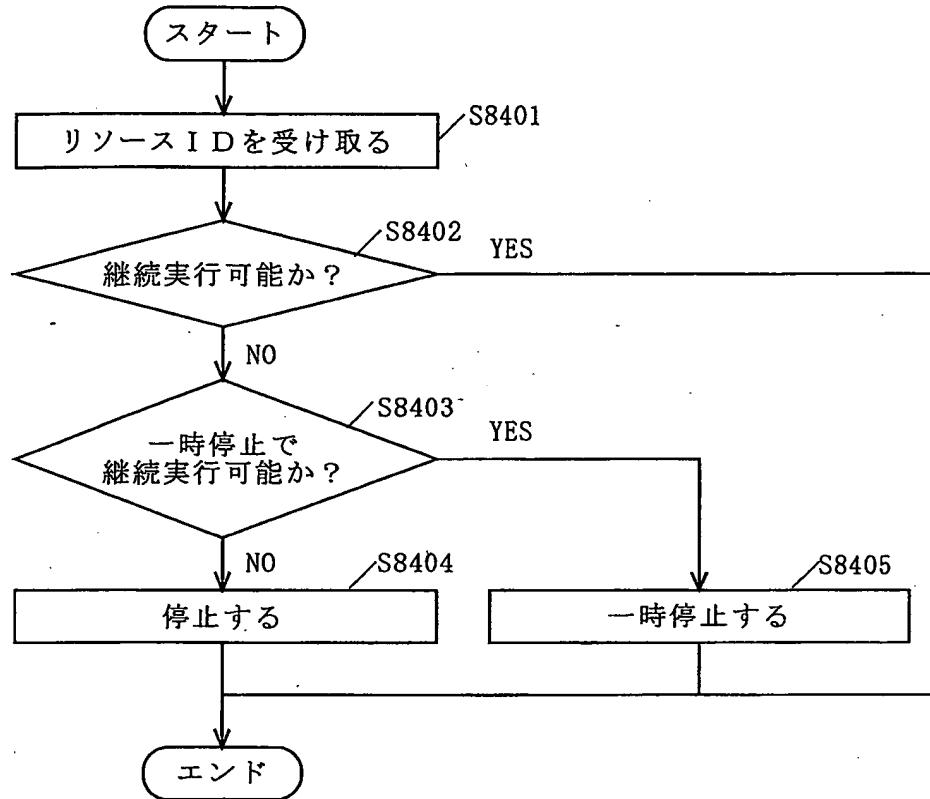


図 8 5

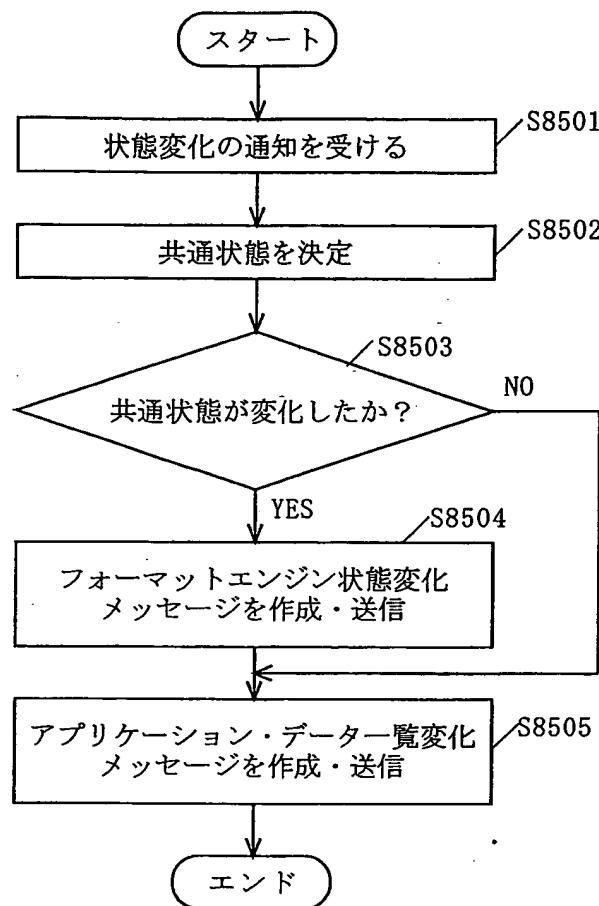


図 8 6

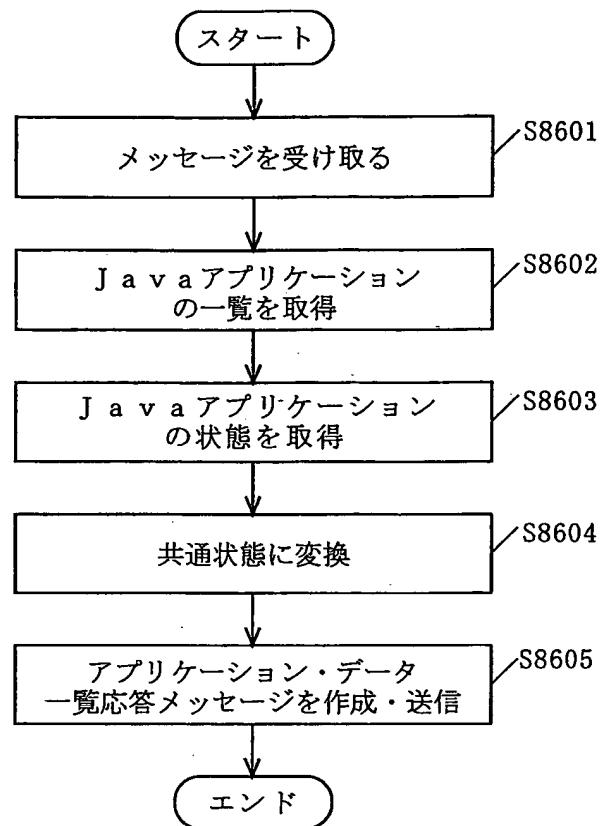


図 8 7

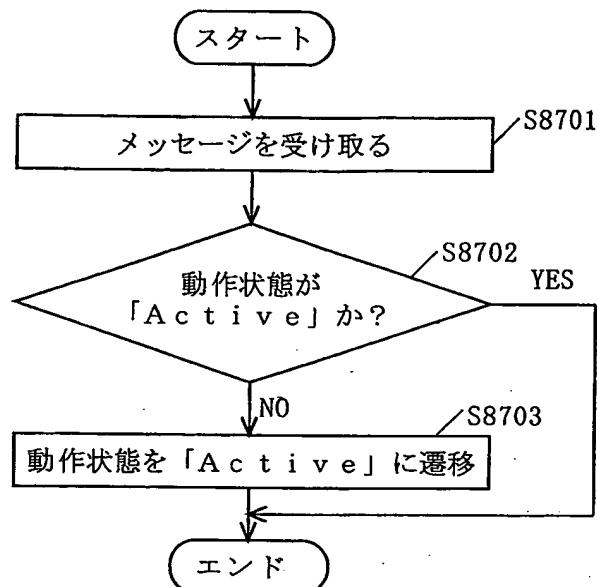


図 8 8

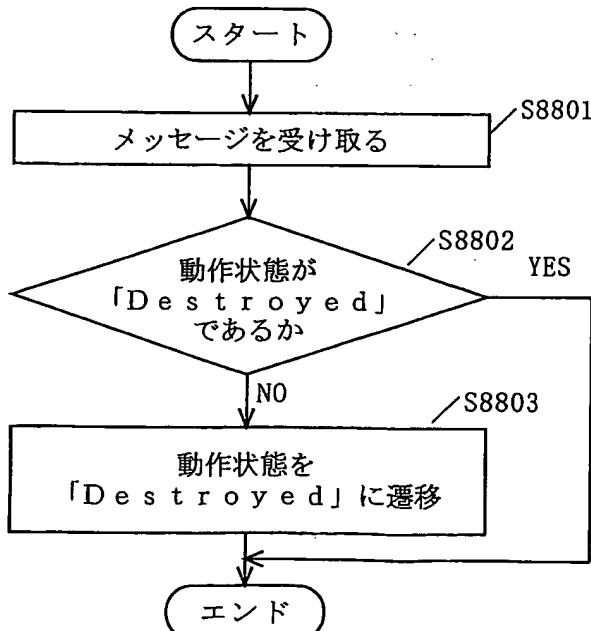


図 8 9

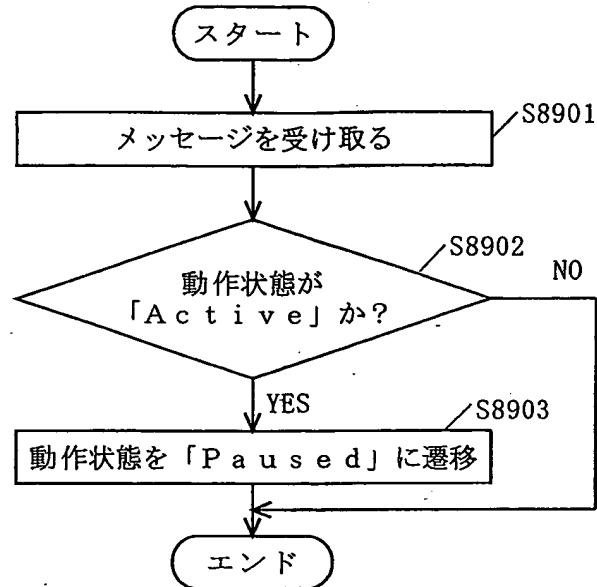


図 9 0

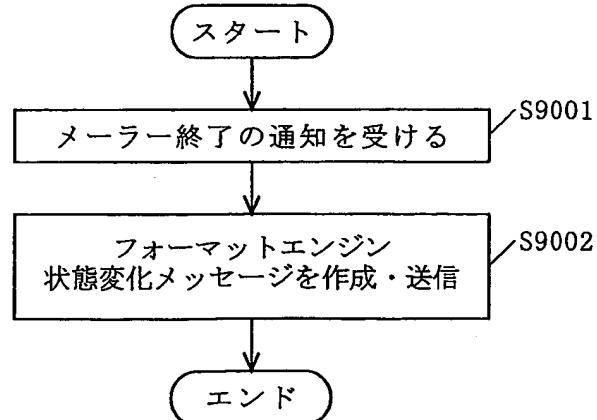


図 9 1

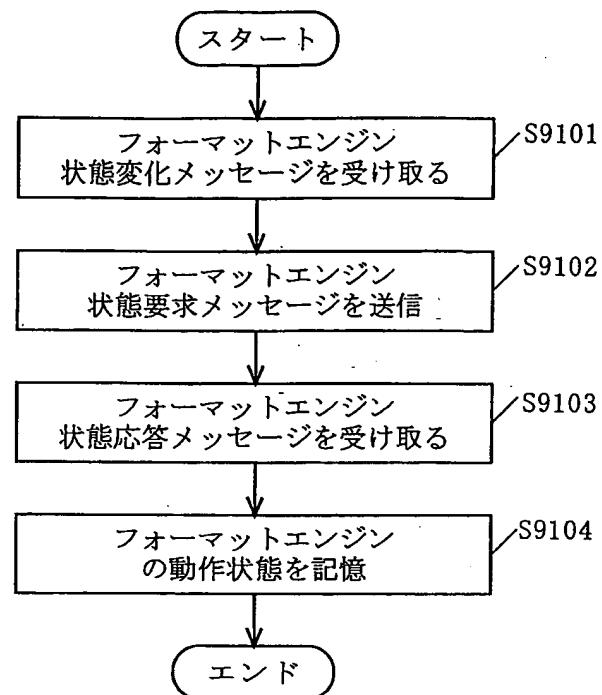


図 9 2

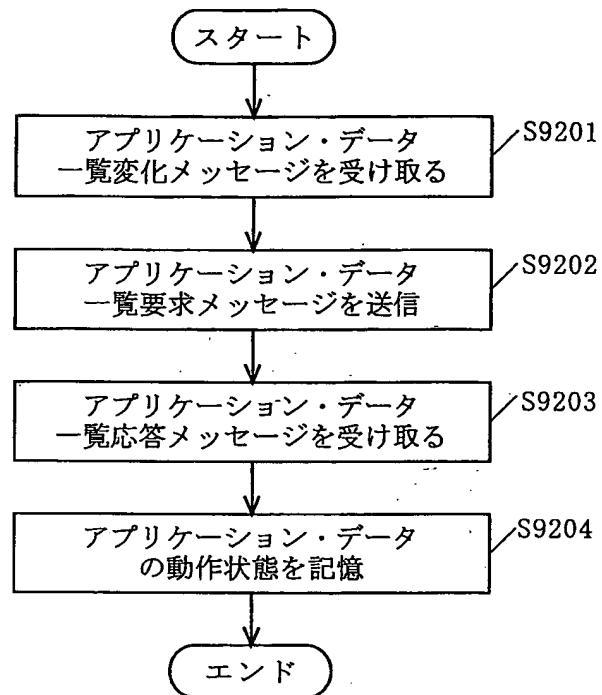


図 9 3

Java ミドルウェア	HTML ブラウザ	メール
停止中	停止中	停止中
一時停止中	停止中	停止中
停止中	一時停止中	停止中
一時停止中	一時停止中	停止中

図 9 4

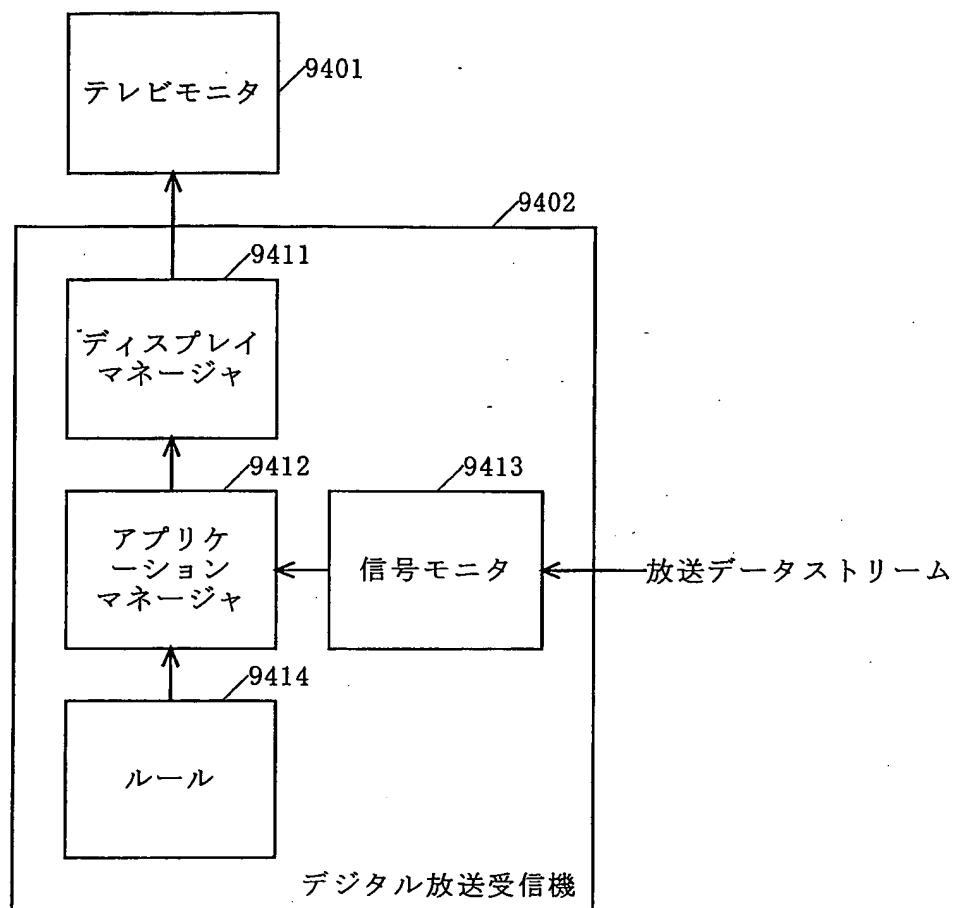
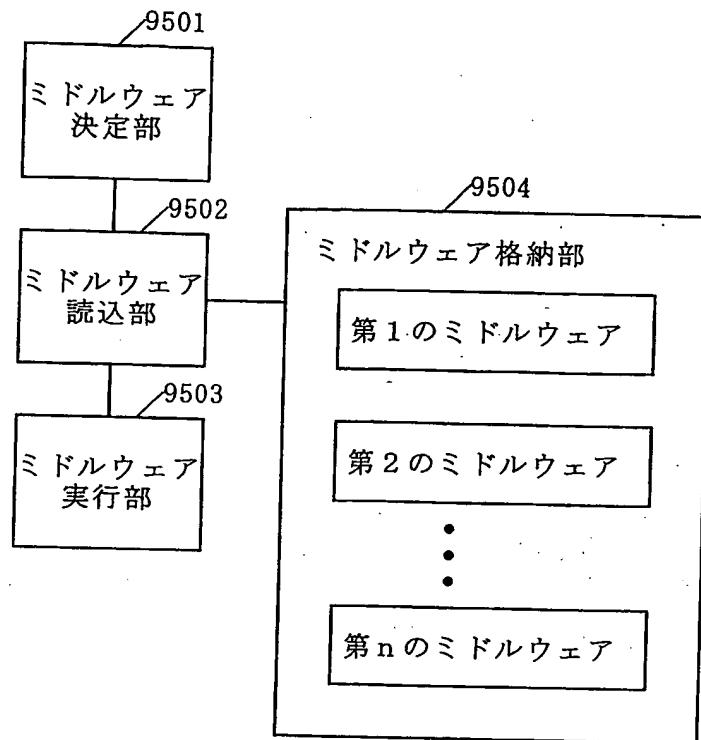


図 9 5



INTERNATIONAL SEARCH REPORT

International application No.

PCT/JP03/12932

A. CLASSIFICATION OF SUBJECT MATTER

Int.Cl⁷ H04N5/44, 5/93

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

Int.Cl⁷ H04N5/38-5/46, 5/91-5/956, 5/78-5/784, G06F3/00, 13/00

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Jitsuyo Shinan Koho 1922-1996 Toroku Jitsuyo Shinan Koho 1994-2003
Kokai Jitsuyo Shinan Koho 1971-2003 Jitsuyo Shinan Toroku Koho 1996-2003

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No...
Y A	JP 7-44477 A (Canon Inc.), 14 February, 1995 (14.02.95), Full text & EP 637157 A & US 5889193 A	1-3, 8 4-7
Y A	JP 11-317991 A (Toshiba Corp.), 16 November, 1999 (16.11.99), Full text & EP 1020858 A & WO 99/18574 A1	1-3, 8 4-7
A	JP 2002-238003 A (Matsushita Electric Industrial Co., Ltd.), 23 August, 2002 (23.08.02), Full text (Family: none)	1-8

 Further documents are listed in the continuation of Box C. See patent family annex.

• Special categories of cited documents:	
"A" document defining the general state of the art which is not considered to be of particular relevance	"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"E" earlier document but published on or after the international filing date	"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)	"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art
"O" document referring to an oral disclosure, use, exhibition or other means	"&" document member of the same patent family
"P" document published prior to the international filing date but later than the priority date claimed	

Date of the actual completion of the international search
22 December, 2003 (22.12.03)Date of mailing of the international search report
20 January, 2004 (20.01.04)Name and mailing address of the ISA/
Japanese Patent Office

Authorized officer

Facsimile No.

Telephone No.

国際調査報告

国際出願番号 PCT/JP03/12932

A. 発明の属する分野の分類 (国際特許分類 (IPC))
Int. c1' H04N5/44, 5/93

B. 調査を行った分野

調査を行った最小限資料 (国際特許分類 (IPC))
Int. c1' H04N5/38-5/46, 5/91-5/956, 5/78-5/784,
G06F3/00, 13/00

最小限資料以外の資料で調査を行った分野に含まれるもの

日本国実用新案公報 1922-1996年
日本国公開実用新案公報 1971-2003年
日本国登録実用新案公報 1994-2003年
日本国実用新案登録公報 1996-2003年

国際調査で使用した電子データベース (データベースの名称、調査に使用した用語)

C. 関連すると認められる文献

引用文献の カテゴリー*	引用文献名 及び一部の箇所が関連するときは、その関連する箇所の表示	関連する 請求の範囲の番号
Y	JP 7-44477 A (キャノン株式会社) 1995.02.1	1-3, 8
A	4, 全文 & EP 637157 A & US 5889193 A	4-7
Y	JP 11-317991 A (株式会社東芝) 1999.11.1	1-3, 8
A	6, 全文 & EP 1020858 A & WO 99/18574 A1	4-7
A	JP 2002-238003 A (松下電器産業株式会社) 20 02.08.23, 全文(ファミリーなし)	1-8

 C欄の続きをにも文献が列挙されている。 パテントファミリーに関する別紙を参照。

* 引用文献のカテゴリー

- 「A」特に関連のある文献ではなく、一般的技術水準を示すもの
- 「E」国際出願日前の出願または特許であるが、国際出願日以後に公表されたもの
- 「L」優先権主張に疑義を提起する文献又は他の文献の発行日若しくは他の特別な理由を確立するために引用する文献 (理由を付す)
- 「O」口頭による開示、使用、展示等に言及する文献
- 「P」国際出願日前で、かつ優先権の主張の基礎となる出願

の日の後に公表された文献
「T」国際出願日又は優先日後に公表された文献であって出願と矛盾するものではなく、発明の原理又は理論の理解のために引用するもの
「X」特に関連のある文献であって、当該文献のみで発明の新規性又は進歩性がないと考えられるもの
「Y」特に関連のある文献であって、当該文献と他の1以上の文献との、当業者にとって自明である組合せによって進歩性がないと考えられるもの
「&」同一パテントファミリー文献

国際調査を完了した日 22. 12. 03	国際調査報告の発送日 20. 1. 2004
国際調査機関の名称及びあて先 日本国特許庁 (ISA/JP) 郵便番号 100-8915 東京都千代田区霞が関三丁目4番3号	特許庁審査官 (権限のある職員) 西谷 慎人 印 電話番号 03-3581-1101 内線 3581 5P 9187

This Page is Inserted by IFW Indexing and Scanning
Operations and is not part of the Official Record

BEST AVAILABLE IMAGES

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images include but are not limited to the items checked:

- BLACK BORDERS**
- IMAGE CUT OFF AT TOP, BOTTOM OR SIDES**
- FADED TEXT OR DRAWING**
- BLURRED OR ILLEGIBLE TEXT OR DRAWING**
- SKEWED/SLANTED IMAGES**
- COLOR OR BLACK AND WHITE PHOTOGRAPHS**
- GRAY SCALE DOCUMENTS**
- LINES OR MARKS ON ORIGINAL DOCUMENT**
- REFERENCE(S) OR EXHIBIT(S) SUBMITTED ARE POOR QUALITY**
- OTHER: _____**

IMAGES ARE BEST AVAILABLE COPY.

As rescanning these documents will not correct the image problems checked, please do not report these problems to the IFW Image Problem Mailbox.